Grounding Recursive Aggregates

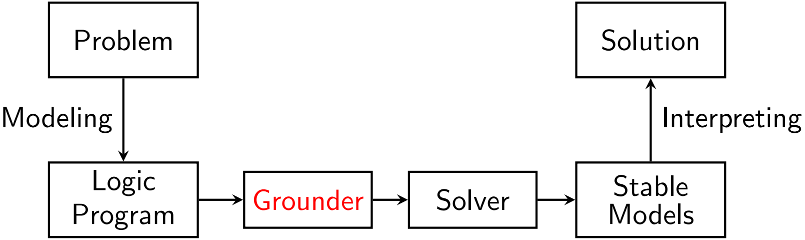
Martin Gebser *Roland Kaminski* Torsten Schaub

University of Potsdam

Outline

Outline

Introduction



Grounding and Solving

- some grounders (in chronological order)
  - `lparse` (grounding using domain predicates)
  - `dlv` (semi-naive evaluation based grounding)
  - `gringo` (semi-naive evaluation based since version 3)

Motivation

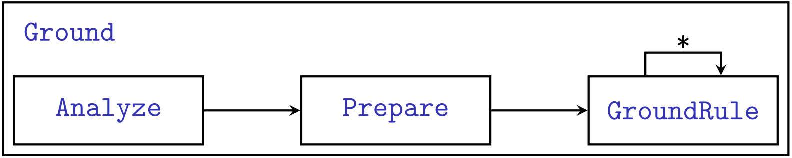
- state restricted aggregate support in ASP grounders
  - `lparse` relies on domain predicates for grounding
    - support for recursive aggregates
    - unhandy for modeling
    - possibly large groundings
  - `dlv` only supports stratified aggregates
    - no support for recursive aggregates
    - additional syntactic restriction on programs
- goal incorporate aggregates into semi-naive grounding
  - only requires programs to be safe
  - implemented in `gringo` series 4

Motivation

- state restricted aggregate support in ASP grounders
  - `lparse` relies on domain predicates for grounding
    - support for recursive aggregates
    - unhandy for modeling
    - possibly large groundings
  - `dlv` only supports stratified aggregates
    - no support for recursive aggregates
    - additional syntactic restriction on programs
- goal incorporate aggregates into semi-naive grounding
  - only requires programs to be safe
  - implemented in `gringo` series 4

Outline

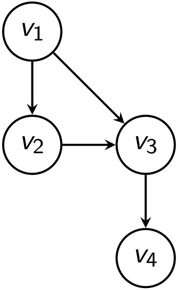
Grounding Normal Logic Programs



- grounding algorithm `Ground` uses three functions

`Analyze`	groups rules into components and determines recursive predicates
`Prepare`	rewrites rules based on recursive predicates
`GroundRule`	instantiates rules iteratively

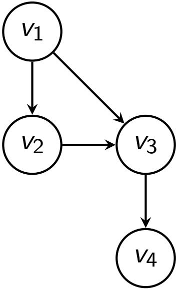
Example: Reachability Problem



$$vertex(v_1). \quad edge(v_1, v_2).$$
$$vertex(v_2). \quad edge(v_1, v_3).$$
$$vertex(v_3). \quad edge(v_2, v_3).$$
$$vertex(v_4). \quad edge(v_3, v_4).$$

$$reach(X, Y) \leftarrow edge(X, Y). \tag{1}$$
$$reach(X, Y) \leftarrow reach(X, Z), edge(Z, Y). \tag{2}$$

Example: Reachability Problem

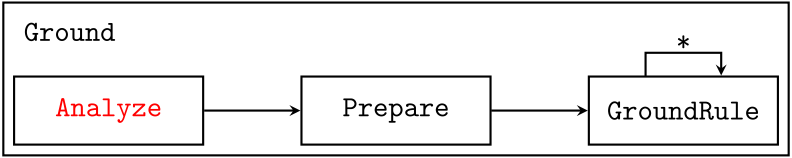


$$\begin{array}{ll} vertex(v_1). & edge(v_1, v_2). \\ vertex(v_2). & edge(v_1, v_3). \\ vertex(v_3). & edge(v_2, v_3). \\ vertex(v_4). & edge(v_3, v_4). \end{array}$$

$$reach(X, Y) \leftarrow edge(X, Y). \tag{1}$$

$$reach(X, Y) \leftarrow reach(X, Z), edge(Z, Y). \tag{2}$$

Grounding Normal Logic Programs



Dependency Graph

- dependency graph $G = (V, E)$ for program P
  - $V = P$
  - $E = \{(r_1, r_2) \in V \times V \mid l \in \mathrm{body}^{\pm}(r_2),$
    $\mathrm{head}(r_1) \text{ unifies } l\}$

  ($\mathrm{body}^{\pm}$ only returns positive symbolic atoms)
- positive dependency graph
  - only considers body^{+} instead of $\mathrm{body}^{\pm}$

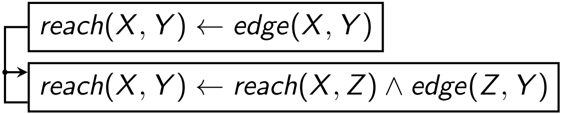
- example

Dependency Graph

- dependency graph $G = (V, E)$ for program P
  - $V = P$
  - $E = \{(r_1, r_2) \in V \times V \mid l \in \mathrm{body}^{\pm}(r_2),$
    $\mathrm{head}(r_1)$ unifies $l\}$

    ($\mathrm{body}^{\pm}$ only returns positive symbolic atoms)
- positive dependency graph
  - only considers body^{+} instead of $\mathrm{body}^{\pm}$

- example



Analyzing Logic Programs

function `Analyze`(P)

  let G be the dependency graph of P

    S be the strongly connected components of G

  $L \leftarrow []$

  foreach C **in** S **do**

    let G^+ be the positive dependency graph of C

      S^+ be the strongly connected components of G^+

    foreach C^+ **in** S^+ **do**

      let $A_\mathrm{r} = \{a \in \mathrm{body}^\pm(r_2) \mid r_1 \in P, r_2 \in C^+, \mathrm{head}(r_1) \text{ unifies } a\}$

      $(L, P) \leftarrow (L + [(C^+, A_\mathrm{r})], P \setminus C^+)$

  return L

Analyzing Logic Programs

function `Analyze`(P)
  let G be the dependency graph of P
    S be the strongly connected components of G
  $L \leftarrow []$
  foreach C **in** S **do**
    let G^+ be the positive dependency graph of C
      S^+ be the strongly connected components of G^+
    foreach C^+ **in** S^+ **do**
      let $A_{\mathrm{r}} = \{a \in \mathrm{body}^{\pm}(r_2) \mid r_1 \in P, r_2 \in C^+, \mathrm{head}(r_1) \text{ unifies } a\}$
      $(L, P) \leftarrow (L + [(C^+, A_{\mathrm{r}})], P \setminus C^+)$
  return L

Analyzing Logic Programs

function `Analyze`(P)
  let G be the dependency graph of P
    S be the strongly connected components of G
  $L \leftarrow []$
  foreach C **in** S **do**
    let G^+ be the positive dependency graph of C
      S^+ be the strongly connected components of G^+
    foreach C^+ **in** S^+ **do**
      let $A_\mathrm{r} = \{a \in \mathrm{body}^\pm(r_2) \mid r_1 \in P, r_2 \in C^+, \mathrm{head}(r_1) \text{ unifies } a\}$
      $(L, P) \leftarrow (L + [(C^+, A_\mathrm{r})], P \setminus C^+)$
  return L

Analyzing Logic Programs

```
function Analyze(P)
    let G be the dependency graph of P
        S be the strongly connected components of G
    L ← []
    foreach C in S do
        let G⁺ be the positive dependency graph of C
            S⁺ be the strongly connected components of G⁺
        foreach C⁺ in S⁺ do
            let A_r = {a ∈ body^±(r_2) | r_1 ∈ P, r_2 ∈ C⁺,
                                           head(r_1) unifies a}
            (L, P) ← (L + [(C⁺, A_r)], P \ C⁺)
    return L
```

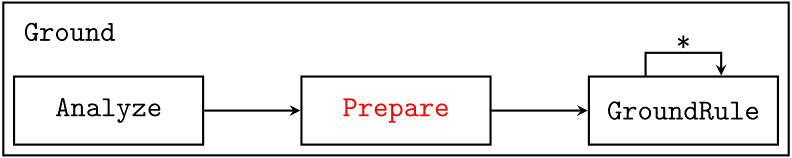
Example: Reachability Encoding Analyzed

Component$_1$: $reach(X, Y) \leftarrow edge(X, Y)$

Component$_2$: $reach(X, Y) \leftarrow \mathbf{reach}(\mathbf{X}, \mathbf{Z}) \wedge edge(Z, Y)$

- only non-factual rules are considered
- atom $reach(X, Z)$ is recursive in the second component

Grounding Normal Logic Programs



Preparing Logic Programs

function $\texttt{Prepare}(C, A_\mathrm{r})$

  $L \leftarrow \emptyset$

  foreach r **in** C **do**

    $D \leftarrow \emptyset$

    foreach $p(\boldsymbol{x})$ **in** $\mathrm{body}^+(r) \cap A_\mathrm{r}$ **do**

$$L \leftarrow L \cup \left\{ \begin{aligned} \mathrm{head}(r) \leftarrow\ & \textstyle\bigwedge_{q(\boldsymbol{y}) \in D} q_\mathrm{o}(\boldsymbol{y}) \wedge p_\mathrm{n}(\boldsymbol{x}) \\ & \wedge \textstyle\bigwedge_{q(\boldsymbol{y}) \in \mathrm{body}^+(r) \setminus (D \cup \{p(\boldsymbol{x})\})} q_\mathrm{a}(\boldsymbol{y}) \\ & \wedge \textstyle\bigwedge_{l \in \mathrm{body}(r) \setminus \mathrm{body}^+(r)} l \end{aligned} \right\}$$

      $D \leftarrow D \cup \{p(\boldsymbol{x})\}$

    if $\mathrm{body}^+(r) \cap A_\mathrm{r} = \emptyset$ **then**

$$L \leftarrow L \cup \left\{ \begin{aligned} \mathrm{head}(r) \leftarrow\ & \textstyle\bigwedge_{p(\boldsymbol{x}) \in \mathrm{body}^+(r)} p_\mathrm{n}(\boldsymbol{x}) \\ & \wedge \textstyle\bigwedge_{l \in \mathrm{body}(r) \setminus \mathrm{body}^+(r)} l \end{aligned} \right\}$$

  return L

Preparing Logic Programs

function $\texttt{Prepare}(C, A_{\mathrm{r}})$

  $L \leftarrow \emptyset$

  foreach r **in** C **do**

    $D \leftarrow \emptyset$

    foreach $p(\boldsymbol{x})$ **in** $\mathrm{body}^+(r) \cap A_{\mathrm{r}}$ **do**

$$L \leftarrow L \cup \left\{ \begin{aligned} \mathrm{head}(r) \leftarrow & \textstyle\bigwedge_{q(\boldsymbol{y}) \in D} q_{\mathrm{o}}(\boldsymbol{y}) \wedge p_{\mathrm{n}}(\boldsymbol{x}) \\ & \wedge \textstyle\bigwedge_{q(\boldsymbol{y}) \in \mathrm{body}^+(r) \setminus (D \cup \{p(\boldsymbol{x})\})} q_{\mathrm{a}}(\boldsymbol{y}) \\ & \wedge \textstyle\bigwedge_{l \in \mathrm{body}(r) \setminus \mathrm{body}^+(r)} l \end{aligned} \right\}$$

      $D \leftarrow D \cup \{p(\boldsymbol{x})\}$

    if $\mathrm{body}^+(r) \cap A_{\mathrm{r}} = \emptyset$ **then**

$$L \leftarrow L \cup \left\{ \begin{aligned} \mathrm{head}(r) \leftarrow & \textstyle\bigwedge_{p(\boldsymbol{x}) \in \mathrm{body}^+(r)} p_{\mathrm{n}}(\boldsymbol{x}) \\ & \wedge \textstyle\bigwedge_{l \in \mathrm{body}(r) \setminus \mathrm{body}^+(r)} l \end{aligned} \right\}$$

  return L

Preparing Logic Programs

function `Prepare`(C, A_r)

    $L \leftarrow \emptyset$

    foreach r **in** C **do**

        $D \leftarrow \emptyset$

        foreach $p(\boldsymbol{x})$ **in** $\mathrm{body}^+(r) \cap A_\mathrm{r}$ **do**

$$L \leftarrow L \cup \left\{ \begin{aligned} \mathrm{head}(r) \leftarrow & \textstyle\bigwedge_{q(\boldsymbol{y}) \in D} q_\mathrm{o}(\boldsymbol{y}) \wedge p_\mathrm{n}(\boldsymbol{x}) \\ & \wedge \textstyle\bigwedge_{q(\boldsymbol{y}) \in \mathrm{body}^+(r) \setminus (D \cup \{p(\boldsymbol{x})\})} q_\mathrm{a}(\boldsymbol{y}) \\ & \wedge \textstyle\bigwedge_{l \in \mathrm{body}(r) \setminus \mathrm{body}^+(r)} l \end{aligned} \right\}$$

            $D \leftarrow D \cup \{p(\boldsymbol{x})\}$

        if $\mathrm{body}^+(r) \cap A_\mathrm{r} = \emptyset$ **then**

$$L \leftarrow L \cup \left\{ \begin{aligned} \mathrm{head}(r) \leftarrow & \textstyle\bigwedge_{p(\boldsymbol{x}) \in \mathrm{body}^+(r)} p_\mathrm{n}(\boldsymbol{x}) \\ & \wedge \textstyle\bigwedge_{l \in \mathrm{body}(r) \setminus \mathrm{body}^+(r)} l \end{aligned} \right\}$$

    return L

Preparing Logic Programs

function $\texttt{Prepare}(C, A_\mathrm{r})$

  $L \leftarrow \emptyset$

  foreach r **in** C **do**

    $D \leftarrow \emptyset$

    foreach $p(\boldsymbol{x})$ **in** $\mathrm{body}^+(r) \cap A_\mathrm{r}$ **do**

$$L \leftarrow L \cup \left\{ \begin{aligned} \mathrm{head}(r) \leftarrow & \textstyle\bigwedge_{q(\boldsymbol{y}) \in D} q_\mathrm{o}(\boldsymbol{y}) \wedge p_\mathrm{n}(\boldsymbol{x}) \\ \wedge & \textstyle\bigwedge_{q(\boldsymbol{y}) \in \mathrm{body}^+(r) \setminus (D \cup \{p(\boldsymbol{x})\})} q_\mathrm{a}(\boldsymbol{y}) \\ \wedge & \textstyle\bigwedge_{l \in \mathrm{body}(r) \setminus \mathrm{body}^+(r)} l \end{aligned} \right\}$$

      $D \leftarrow D \cup \{p(\boldsymbol{x})\}$

    if $\mathrm{body}^+(r) \cap A_\mathrm{r} = \emptyset$ **then**

$$L \leftarrow L \cup \left\{ \begin{aligned} \mathrm{head}(r) \leftarrow & \textstyle\bigwedge_{p(\boldsymbol{x}) \in \mathrm{body}^+(r)} p_\mathrm{n}(\boldsymbol{x}) \\ \wedge & \textstyle\bigwedge_{l \in \mathrm{body}(r) \setminus \mathrm{body}^+(r)} l \end{aligned} \right\}$$

  return L

Example: Reachability Encoding Prepared

Component$_1$: $reach(X, Y) \leftarrow edge_{\mathrm{n}}(X, Y)$

Component$_2$: $reach(X, Y) \leftarrow \mathbf{reach}_{\mathrm{n}}(\mathbf{X}, \mathbf{Z}) \wedge edge_{\mathrm{a}}(Z, Y)$

- subscript n in first component because there is no recursion
- subscripts n and a in second component
  - recursive atom $\mathbf{reach}(\mathbf{X}, \mathbf{Z})$ is adorned with n
  - non-recursive atom $edge(Z, Y)$ is adorned with a

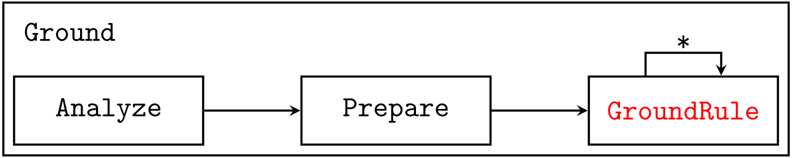
Example: Reachability Encoding Prepared

Component$_1$: $\mathit{reach}(X, Y) \leftarrow \mathit{edge}_{\mathrm{n}}(X, Y)$

Component$_2$: $\mathit{reach}(X, Y) \leftarrow \mathbf{reach}_{\mathrm{n}}(\mathbf{X}, \mathbf{Z}) \wedge \mathit{edge}_{\mathrm{a}}(Z, Y)$

- subscript n in first component because there is no recursion
- subscripts n and a in second component
  - recursive atom $\mathbf{reach}(\mathbf{X}, \mathbf{Z})$ is adorned with n
  - non-recursive atom $\mathit{edge}(Z, Y)$ is adorned with a

Grounding Normal Logic Programs



Safe Body Order and Matches

- safe body order $\mathrm{order}(\{b_1, \ldots, b_n\})$
  - $(b_1, \ldots, b_n)$ is a safe order
    if $\{b_1, \ldots, b_i\}$ is safe for each $1 \leq i \leq n$.
- example
  - $(p(X), \sim q(X))$ is a safe body order
  - $(\sim q(X), p(X))$ is not a safe body order
- $\mathrm{matches}(a, \sigma, A)$
  - symbolic atom a, substitution σ, set of ground atoms A
  - $\subseteq$-minimal substitutions σ' such that $a\sigma' \in A$ and $\sigma \subseteq \sigma'$.
- example
  - $\mathrm{matches}(p(X, Y), \{Y \mapsto a\}, \{p(a, a), p(b, b), p(c, a)\})$
    yields $\{X \mapsto a, Y \mapsto a\}$ and $\{X \mapsto c, Y \mapsto a\}$.

Safe Body Order and Matches

- safe body order $\mathrm{order}(\{b_1, \dots, b_n\})$
  - $(b_1, \dots, b_n)$ is a safe order
    if $\{b_1, \dots, b_i\}$ is safe for each $1 \leq i \leq n$.
- example
  - $(p(X), {\sim}q(X))$ is a safe body order
  - $({\sim}q(X), p(X))$ is not a safe body order
- $\mathrm{matches}(a, \sigma, A)$
  - symbolic atom a, substitution σ, set of ground atoms A
  - $\subseteq$-minimal substitutions σ' such that $a\sigma' \in A$ and $\sigma \subseteq \sigma'$.
- example
  - $\mathrm{matches}(p(X, Y), \{Y \mapsto a\}, \{p(a, a), p(b, b), p(c, a)\})$
    yields $\{X \mapsto a, Y \mapsto a\}$ and $\{X \mapsto c, Y \mapsto a\}$.

Safe Body Order and Matches

- safe body order $\mathrm{order}(\{b_1, \ldots, b_n\})$
  - $(b_1, \ldots, b_n)$ is a safe order
    if $\{b_1, \ldots, b_i\}$ is safe for each $1 \leq i \leq n$.
- example
  - $(p(X), \sim q(X))$ is a safe body order
  - $(\sim q(X), p(X))$ is not a safe body order
- $\mathrm{matches}(a, \sigma, A)$
  - symbolic atom a, substitution σ, set of ground atoms A
  - $\subseteq$-minimal substitutions σ' such that $a\sigma' \in A$ and $\sigma \subseteq \sigma'$.
- example
  - $\mathrm{matches}(p(X, Y), \{Y \mapsto a\}, \{p(a, a), p(b, b), p(c, a)\})$
    yields $\{X \mapsto a, Y \mapsto a\}$ and $\{X \mapsto c, Y \mapsto a\}$.

Safe Body Order and Matches

- safe body order $\mathrm{order}(\{b_1, \ldots, b_n\})$
  - $(b_1, \ldots, b_n)$ is a safe order
    if $\{b_1, \ldots, b_i\}$ is safe for each $1 \leq i \leq n$.
- example
  - $(p(X), {\sim}q(X))$ is a safe body order
  - $({\sim}q(X), p(X))$ is not a safe body order
- $\mathrm{matches}(a, \sigma, A)$
  - symbolic atom a, substitution σ, set of ground atoms A
  - $\subseteq$-minimal substitutions σ' such that $a\sigma' \in A$ and $\sigma \subseteq \sigma'$.
- example
  - $\mathrm{matches}(p(X, Y), \{Y \mapsto a\}, \{p(a, a), p(b, b), p(c, a)\})$
    yields $\{X \mapsto a, Y \mapsto a\}$ and $\{X \mapsto c, Y \mapsto a\}$.

Grounding Rules

```
function GroundRule(r, A_r, A_n, A_o, A_a, A_f)
    let r′ be the original version of rule r
    G ← ∅
    function GroundRule′(B, (b_1, ..., b_n), σ)
        if n = 0 then                                                    // rule instance
            if head(r)σ ∉ A_f then
                G ← G ∪ {head(r)σ ← B}
                A_f ← A_f ∪ {head(r)σ | B = ∅}
        else if b_1 = p_x(x) for x ∈ {o, n, a} then                      // positive literals
            foreach σ′ ∈ matches(p(x), σ, A_x) do
                B′ ← B ∪ {p(x)σ′ | r′ ⊁ p(x), p(x)σ′ ∉ A_f}
                GroundRule′(B′, (b_2, ..., b_n), σ′)
        else if b_1 = ∼a then                                            // negative literals
            if aσ ∉ A_f then
                B′ ← B ∪ {b_1σ | r′ ⊁ b_1, a ∈ A_r or aσ ∈ A_a}
                GroundRule′(B′, (b_2, ..., b_n), σ)
        else                                                             // comparison atoms
            if b_1σ is true then
                GroundRule′(B, (b_2, ..., b_n), σ)
    GroundRule′(∅, order(body(r)), ∅)
    return (G, A_f)
```

Grounding Rules

```
function GroundRule(r, A_r, A_n, A_o, A_a, A_f)
    let r' be the original version of rule r
    G ← ∅
    function GroundRule'(B, (b_1, ..., b_n), σ)
        if n = 0 then                                                   // rule instance
            if head(r)σ ∉ A_f then
                G ← G ∪ {head(r)σ ← B}
                A_f ← A_f ∪ {head(r)σ | B = ∅}
        else if b_1 = p_x(x) for x ∈ {o, n, a} then                     // positive literals
            foreach σ' ∈ matches(p(x), σ, A_x) do
                B' ← B ∪ {p(x)σ' | r' ⊬ p(x), p(x)σ' ∉ A_f}
                GroundRule'(B', (b_2, ..., b_n), σ')
        else if b_1 = ∼a then                                           // negative literals
            if aσ ∉ A_f then
                B' ← B ∪ {b_1σ | r' ⊬ b_1, a ∈ A_r or aσ ∈ A_a}
                GroundRule'(B', (b_2, ..., b_n), σ)
        else                                                            // comparison atoms
            if b_1σ is true then
                GroundRule'(B, (b_2, ..., b_n), σ)
    GroundRule'(∅, order(body(r)), ∅)
    return (G, A_f)
```

Grounding Rules

function GroundRule$(r, A_\mathrm{r}, A_\mathrm{n}, A_\mathrm{o}, A_\mathrm{a}, A_\mathrm{f})$
  let r' be the original version of rule r
  $G \leftarrow \emptyset$
  function GroundRule$'(B, (b_1, \ldots, b_n), \sigma)$
    if $n = 0$ **then** // rule instance
      if $\mathrm{head}(r)\sigma \notin A_\mathrm{f}$ **then**
        $G \leftarrow G \cup \{\mathrm{head}(r)\sigma \leftarrow B\}$
        $A_\mathrm{f} \leftarrow A_\mathrm{f} \cup \{\mathrm{head}(r)\sigma \mid B = \emptyset\}$
    else if $b_1 = p_x(\mathbf{x})$ *for* $x \in \{\mathrm{o}, \mathrm{n}, \mathrm{a}\}$ **then** // positive literals
      foreach $\sigma' \in \mathrm{matches}(p(\mathbf{x}), \sigma, A_x)$ **do**
        $B' \leftarrow B \cup \{p(\mathbf{x})\sigma' \mid r' \not\prec p(\mathbf{x}), p(\mathbf{x})\sigma' \notin A_\mathrm{f}\}$
        GroundRule$'(B', (b_2, \ldots, b_n), \sigma')$
    else if $b_1 = {\sim}a$ **then** // negative literals
      if $a\sigma \notin A_\mathrm{f}$ **then**
        $B' \leftarrow B \cup \{b_1\sigma \mid r' \not\prec b_1, a \in A_\mathrm{r}$ **or** $a\sigma \in A_\mathrm{a}\}$
        GroundRule$'(B', (b_2, \ldots, b_n), \sigma)$
    else // comparison atoms
      if $b_1\sigma$ *is true* **then**
        GroundRule$'(B, (b_2, \ldots, b_n), \sigma)$
  GroundRule$'(\emptyset, \mathrm{order}(\mathrm{body}(r)), \emptyset)$
  return (G, A_f)

Grounding Rules

```
function GroundRule(r, A_r, A_n, A_o, A_a, A_f)
    let r' be the original version of rule r
    G ← ∅
    function GroundRule'(B, (b_1, ..., b_n), σ)
        if n = 0 then                                                  // rule instance
            if head(r)σ ∉ A_f then
                G ← G ∪ {head(r)σ ← B}
                A_f ← A_f ∪ {head(r)σ | B = ∅}
        else if b_1 = p_x(x) for x ∈ {o, n, a} then                    // positive literals
            foreach σ' ∈ matches(p(x), σ, A_x) do
                B' ← B ∪ {p(x)σ' | r' ⊁ p(x), p(x)σ' ∉ A_f}
                GroundRule'(B', (b_2, ..., b_n), σ')
        else if b_1 = ∼a then                                          // negative literals
            if aσ ∉ A_f then
                B' ← B ∪ {b_1σ | r' ⊁ b_1, a ∈ A_r or aσ ∈ A_a}
                GroundRule'(B', (b_2, ..., b_n), σ)
        else                                                           // comparison atoms
            if b_1σ is true then
                GroundRule'(B, (b_2, ..., b_n), σ)
    GroundRule'(∅, order(body(r)), ∅)
    return (G, A_f)
```

Grounding Rules

function GroundRule($r, A_{\mathrm{r}}, A_{\mathrm{n}}, A_{\mathrm{o}}, A_{\mathrm{a}}, A_{\mathrm{f}}$)
  let r' be the original version of rule r
  $G \leftarrow \emptyset$
  function GroundRule$'$($B, (b_1, \ldots, b_n), \sigma$)
    if $n = 0$ **then** // rule instance
      if $\mathrm{head}(r)\sigma \notin A_{\mathrm{f}}$ **then**
        $G \leftarrow G \cup \{\mathrm{head}(r)\sigma \leftarrow B\}$
        $A_{\mathrm{f}} \leftarrow A_{\mathrm{f}} \cup \{\mathrm{head}(r)\sigma \mid B = \emptyset\}$
    else if $b_1 = p_x(\boldsymbol{x})$ *for* $x \in \{\mathrm{o}, \mathrm{n}, \mathrm{a}\}$ **then** // positive literals
      foreach $\sigma' \in \mathrm{matches}(p(\boldsymbol{x}), \sigma, A_x)$ **do**
        $B' \leftarrow B \cup \{p(\boldsymbol{x})\sigma' \mid r' \not\vdash p(\boldsymbol{x}), p(\boldsymbol{x})\sigma' \notin A_{\mathrm{f}}\}$
        GroundRule$'$($B', (b_2, \ldots, b_n), \sigma'$)
    else if $b_1 = {\sim}a$ **then** // negative literals
      if $a\sigma \notin A_{\mathrm{f}}$ **then**
        $B' \leftarrow B \cup \{b_1\sigma \mid r' \not\vdash b_1, a \in A_{\mathrm{r}}$ **or** $a\sigma \in A_{\mathrm{a}}\}$
        GroundRule$'$($B', (b_2, \ldots, b_n), \sigma$)
    else // comparison atoms
      if $b_1\sigma$ *is true* **then**
        GroundRule$'$($B, (b_2, \ldots, b_n), \sigma$)
  GroundRule$'$($\emptyset, \mathrm{order}(\mathrm{body}(r)), \emptyset$)
  return (G, A_{f})

Example: Grounding the First Rule

$$\frac{edge_{\mathrm{n}}(X, Y) \qquad reach(X, Y) \qquad A_{\mathrm{r}} = \emptyset}{}$$

$$\begin{array}{ccc}
edge(v_1, v_2) & \longrightarrow & reach(v_1, v_2) \\
| & & \\
edge(v_1, v_3) & \longrightarrow & reach(v_1, v_3) \\
| & & \\
edge(v_2, v_3) & \longrightarrow & reach(v_2, v_3) \\
| & & \\
edge(v_3, v_4) & \longrightarrow & reach(v_3, v_4)
\end{array}$$

$$A_{\mathrm{f}} = \left\{ \begin{array}{l} edge(v_1, v_2), \\ edge(v_1, v_3), \\ edge(v_2, v_3), \\ edge(v_3, v_4), \\ \ldots \end{array} \right\}$$

$$A_{\mathrm{o}} = \emptyset$$
$$A_{\mathrm{n}} = A_{\mathrm{f}}$$
$$A_{\mathrm{a}} = A_{\mathrm{f}}$$

Example: Grounding the First Rule

$$edge_{\mathrm{n}}(X, Y) \qquad reach(X, Y) \qquad A_{\mathrm{r}} = \emptyset$$

$$\begin{array}{lcl}
\color{red}{edge(v_1, v_2)} & \longrightarrow & reach(v_1, v_2) \\
\vert & & \\
\color{red}{edge(v_1, v_3)} & \longrightarrow & reach(v_1, v_3) \\
\vert & & \\
\color{red}{edge(v_2, v_3)} & \longrightarrow & reach(v_2, v_3) \\
\vert & & \\
\color{red}{edge(v_3, v_4)} & \longrightarrow & reach(v_3, v_4)
\end{array}$$

$$A_{\mathrm{f}} = \left\{ \begin{array}{l} edge(v_1, v_2), \\ edge(v_1, v_3), \\ edge(v_2, v_3), \\ edge(v_3, v_4), \\ \ldots \end{array} \right\}$$

$$A_{\mathrm{o}} = \emptyset$$

$$A_{\mathrm{n}} = A_{\mathrm{f}}$$

$$A_{\mathrm{a}} = A_{\mathrm{f}}$$

Example: Grounding the First Rule

$$\frac{edge_{\mathrm{n}}(X, Y) \qquad reach(X, Y) \qquad A_{\mathrm{r}} = \emptyset}{}$$

$edge(v_1, v_2) \longrightarrow reach(v_1, v_2)$

$|$

$edge(v_1, v_3) \longrightarrow reach(v_1, v_3)$

$|$

$edge(v_2, v_3) \longrightarrow reach(v_2, v_3)$

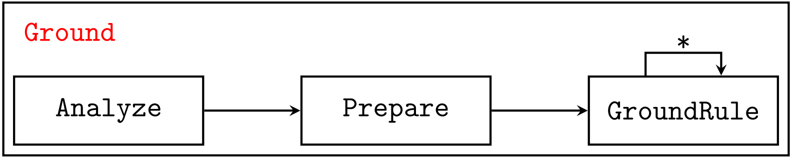
$|$

$edge(v_3, v_4) \longrightarrow reach(v_3, v_4)$

$$A_{\mathrm{f}} = \left\{ \begin{array}{l} edge(v_1, v_2), \\ edge(v_1, v_3), \\ edge(v_2, v_3), \\ edge(v_3, v_4), \\ \ldots \end{array} \right\}$$

$A_{\mathrm{o}} = \emptyset$
$A_{\mathrm{n}} = A_{\mathrm{f}}$
$A_{\mathrm{a}} = A_{\mathrm{f}}$

Grounding Normal Logic Programs



Grounding Normal Logic Programs

function $\texttt{Ground}(P, A_\mathrm{f})$

  $(P_\mathrm{g}, A_\mathrm{a}) \leftarrow (\emptyset, A_\mathrm{f})$

  foreach (C, A_r) **in** $\texttt{Analyze}(P)$ **do**

    $(A_\mathrm{n}, A_\mathrm{o}) \leftarrow (A_\mathrm{a}, \emptyset)$

    repeat

      $A_\Delta \leftarrow \emptyset$

      foreach r **in** $\texttt{Prepare}(C, A_\mathrm{r})$ **do**

        $(P'_\mathrm{g}, A_\mathrm{f}) \leftarrow \texttt{GroundRule}(r, A_\mathrm{r}, A_\mathrm{n}, A_\mathrm{o}, A_\mathrm{a}, A_\mathrm{f})$

        $(A_\Delta, P_\mathrm{g}) \leftarrow (A_\Delta \cup \{\mathrm{head}(r_\mathrm{g}) \mid r_\mathrm{g} \in P'_\mathrm{g}\}, P_\mathrm{g} \cup P'_\mathrm{g})$

      $(A_\mathrm{n}, A_\mathrm{o}, A_\mathrm{a}) \leftarrow (A_\Delta \setminus A_\mathrm{a}, A_\mathrm{a}, A_\Delta \cup A_\mathrm{a})$

    until $A_\mathrm{n} = \emptyset$ **or** $\{r \in C \mid \mathrm{body}^+(r) \cap A_\mathrm{r} \neq \emptyset\} = \emptyset$

  return P_g

Grounding Normal Logic Programs

function $\texttt{Ground}(P, A_{\mathrm{f}})$
  $(P_{\mathrm{g}}, A_{\mathrm{a}}) \leftarrow (\emptyset, A_{\mathrm{f}})$
  foreach (C, A_{r}) **in** $\texttt{Analyze}(P)$ **do**
    $(A_{\mathrm{n}}, A_{\mathrm{o}}) \leftarrow (A_{\mathrm{a}}, \emptyset)$
    repeat
      $A_{\Delta} \leftarrow \emptyset$
      foreach r **in** $\texttt{Prepare}(C, A_{\mathrm{r}})$ **do**
        $(P'_{\mathrm{g}}, A_{\mathrm{f}}) \leftarrow \texttt{GroundRule}(r, A_{\mathrm{r}}, A_{\mathrm{n}}, A_{\mathrm{o}}, A_{\mathrm{a}}, A_{\mathrm{f}})$
        $(A_{\Delta}, P_{\mathrm{g}}) \leftarrow (A_{\Delta} \cup \{\mathrm{head}(r_{\mathrm{g}}) \mid r_{\mathrm{g}} \in P'_{\mathrm{g}}\}, P_{\mathrm{g}} \cup P'_{\mathrm{g}})$
      $(A_{\mathrm{n}}, A_{\mathrm{o}}, A_{\mathrm{a}}) \leftarrow (A_{\Delta} \setminus A_{\mathrm{a}}, A_{\mathrm{a}}, A_{\Delta} \cup A_{\mathrm{a}})$
    until $A_{\mathrm{n}} = \emptyset$ **or** $\{r \in C \mid \mathrm{body}^{+}(r) \cap A_{\mathrm{r}} \neq \emptyset\} = \emptyset$
  return P_{g}

Grounding Normal Logic Programs

function $\texttt{Ground}(P, A_{\mathrm{f}})$

  $(P_{\mathrm{g}}, A_{\mathrm{a}}) \leftarrow (\emptyset, A_{\mathrm{f}})$

  foreach (C, A_{r}) **in** $\texttt{Analyze}(P)$ **do**

    $(A_{\mathrm{n}}, A_{\mathrm{o}}) \leftarrow (A_{\mathrm{a}}, \emptyset)$

    repeat

      $A_{\Delta} \leftarrow \emptyset$

      foreach r **in** $\texttt{Prepare}(C, A_{\mathrm{r}})$ **do**

        $(P'_{\mathrm{g}}, A_{\mathrm{f}}) \leftarrow \texttt{GroundRule}(r, A_{\mathrm{r}}, A_{\mathrm{n}}, A_{\mathrm{o}}, A_{\mathrm{a}}, A_{\mathrm{f}})$

        $(A_{\Delta}, P_{\mathrm{g}}) \leftarrow (A_{\Delta} \cup \{\mathrm{head}(r_{\mathrm{g}}) \mid r_{\mathrm{g}} \in P'_{\mathrm{g}}\}, P_{\mathrm{g}} \cup P'_{\mathrm{g}})$

      $(A_{\mathrm{n}}, A_{\mathrm{o}}, A_{\mathrm{a}}) \leftarrow (A_{\Delta} \setminus A_{\mathrm{a}}, A_{\mathrm{a}}, A_{\Delta} \cup A_{\mathrm{a}})$

    until $A_{\mathrm{n}} = \emptyset$ **or** $\{r \in C \mid \mathrm{body}^{+}(r) \cap A_{\mathrm{r}} \neq \emptyset\} = \emptyset$

  return P_{g}

Grounding Normal Logic Programs

function $\texttt{Ground}(P, A_\mathrm{f})$
$(P_\mathrm{g}, A_\mathrm{a}) \leftarrow (\emptyset, A_\mathrm{f})$
foreach (C, A_r) **in** $\texttt{Analyze}(P)$ **do**
$(A_\mathrm{n}, A_\mathrm{o}) \leftarrow (A_\mathrm{a}, \emptyset)$
repeat
$A_\Delta \leftarrow \emptyset$
foreach r **in** $\texttt{Prepare}(C, A_\mathrm{r})$ **do**
$(P'_\mathrm{g}, A_\mathrm{f}) \leftarrow \texttt{GroundRule}(r, A_\mathrm{r}, A_\mathrm{n}, A_\mathrm{o}, A_\mathrm{a}, A_\mathrm{f})$
$(A_\Delta, P_\mathrm{g}) \leftarrow (A_\Delta \cup \{\mathrm{head}(r_\mathrm{g}) \mid r_\mathrm{g} \in P'_\mathrm{g}\}, P_\mathrm{g} \cup P'_\mathrm{g})$
$(A_\mathrm{n}, A_\mathrm{o}, A_\mathrm{a}) \leftarrow (A_\Delta \setminus A_\mathrm{a}, A_\mathrm{a}, A_\Delta \cup A_\mathrm{a})$
until $A_\mathrm{n} = \emptyset$ **or** $\{r \in C \mid \mathrm{body}^+(r) \cap A_\mathrm{r} \neq \emptyset\} = \emptyset$
return P_g

Grounding Normal Logic Programs

function $\texttt{Ground}(P, A_{\mathrm{f}})$
$\quad (P_{\mathrm{g}}, A_{\mathrm{a}}) \leftarrow (\emptyset, A_{\mathrm{f}})$
$\quad$ **foreach** (C, A_{r}) **in** $\texttt{Analyze}(P)$ **do**
$\quad\quad (A_{\mathrm{n}}, A_{\mathrm{o}}) \leftarrow (A_{\mathrm{a}}, \emptyset)$
$\quad\quad$ **repeat**
$\quad\quad\quad A_{\Delta} \leftarrow \emptyset$
$\quad\quad\quad$ **foreach** r **in** $\texttt{Prepare}(C, A_{\mathrm{r}})$ **do**
$\quad\quad\quad\quad (P'_{\mathrm{g}}, A_{\mathrm{f}}) \leftarrow \texttt{GroundRule}(r, A_{\mathrm{r}}, A_{\mathrm{n}}, A_{\mathrm{o}}, A_{\mathrm{a}}, A_{\mathrm{f}})$
$\quad\quad\quad\quad (A_{\Delta}, P_{\mathrm{g}}) \leftarrow (A_{\Delta} \cup \{\mathrm{head}(r_{\mathrm{g}}) \mid r_{\mathrm{g}} \in P'_{\mathrm{g}}\}, P_{\mathrm{g}} \cup P'_{\mathrm{g}})$
$\quad\quad\quad (A_{\mathrm{n}}, A_{\mathrm{o}}, A_{\mathrm{a}}) \leftarrow (A_{\Delta} \setminus A_{\mathrm{a}}, A_{\mathrm{a}}, A_{\Delta} \cup A_{\mathrm{a}})$
$\quad\quad$ **until** $A_{\mathrm{n}} = \emptyset$ **or** $\{r \in C \mid \mathrm{body}^{+}(r) \cap A_{\mathrm{r}} \neq \emptyset\} = \emptyset$
$\quad$ **return** P_{g}

Grounding Normal Logic Programs

function $\texttt{Ground}(P, A_\mathrm{f})$
  $(P_\mathrm{g}, A_\mathrm{a}) \leftarrow (\emptyset, A_\mathrm{f})$
  foreach (C, A_r) **in** $\texttt{Analyze}(P)$ **do**
    $(A_\mathrm{n}, A_\mathrm{o}) \leftarrow (A_\mathrm{a}, \emptyset)$
    repeat
      $A_\Delta \leftarrow \emptyset$
      foreach r **in** $\texttt{Prepare}(C, A_\mathrm{r})$ **do**
        $(P'_\mathrm{g}, A_\mathrm{f}) \leftarrow \texttt{GroundRule}(r, A_\mathrm{r}, A_\mathrm{n}, A_\mathrm{o}, A_\mathrm{a}, A_\mathrm{f})$
        $(A_\Delta, P_\mathrm{g}) \leftarrow (A_\Delta \cup \{\mathrm{head}(r_\mathrm{g}) \mid r_\mathrm{g} \in P'_\mathrm{g}\}, P_\mathrm{g} \cup P'_\mathrm{g})$
      $(A_\mathrm{n}, A_\mathrm{o}, A_\mathrm{a}) \leftarrow (A_\Delta \setminus A_\mathrm{a}, A_\mathrm{a}, A_\Delta \cup A_\mathrm{a})$
    until $A_\mathrm{n} = \emptyset$ **or** $\{r \in C \mid \mathrm{body}^+(r) \cap A_\mathrm{r} \neq \emptyset\} = \emptyset$
  return P_g

Grounding Normal Logic Programs

function $\texttt{Ground}(P, A_{\mathrm{f}})$

  $(P_{\mathrm{g}}, A_{\mathrm{a}}) \leftarrow (\emptyset, A_{\mathrm{f}})$

  foreach (C, A_{r}) **in** $\texttt{Analyze}(P)$ **do**

    $(A_{\mathrm{n}}, A_{\mathrm{o}}) \leftarrow (A_{\mathrm{a}}, \emptyset)$

    repeat

      $A_{\Delta} \leftarrow \emptyset$

      foreach r **in** $\texttt{Prepare}(C, A_{\mathrm{r}})$ **do**

        $(P'_{\mathrm{g}}, A_{\mathrm{f}}) \leftarrow \texttt{GroundRule}(r, A_{\mathrm{r}}, A_{\mathrm{n}}, A_{\mathrm{o}}, A_{\mathrm{a}}, A_{\mathrm{f}})$

        $(A_{\Delta}, P_{\mathrm{g}}) \leftarrow (A_{\Delta} \cup \{\mathrm{head}(r_{\mathrm{g}}) \mid r_{\mathrm{g}} \in P'_{\mathrm{g}}\}, P_{\mathrm{g}} \cup P'_{\mathrm{g}})$

      $(A_{\mathrm{n}}, A_{\mathrm{o}}, A_{\mathrm{a}}) \leftarrow (A_{\Delta} \setminus A_{\mathrm{a}}, A_{\mathrm{a}}, A_{\Delta} \cup A_{\mathrm{a}})$

    until $A_{\mathrm{n}} = \emptyset$ **or** $\{r \in C \mid \mathrm{body}^{+}(r) \cap A_{\mathrm{r}} \neq \emptyset\} = \emptyset$

  return P_{g}

Grounding Normal Logic Programs

function $\texttt{Ground}(P, A_{\mathrm{f}})$
    $(P_{\mathrm{g}}, A_{\mathrm{a}}) \leftarrow (\emptyset, A_{\mathrm{f}})$
    foreach (C, A_{r}) **in** $\texttt{Analyze}(P)$ **do**
        $(A_{\mathrm{n}}, A_{\mathrm{o}}) \leftarrow (A_{\mathrm{a}}, \emptyset)$
        repeat
            $A_{\Delta} \leftarrow \emptyset$
            foreach r **in** $\texttt{Prepare}(C, A_{\mathrm{r}})$ **do**
                $(P'_{\mathrm{g}}, A_{\mathrm{f}}) \leftarrow \texttt{GroundRule}(r, A_{\mathrm{r}}, A_{\mathrm{n}}, A_{\mathrm{o}}, A_{\mathrm{a}}, A_{\mathrm{f}})$
                $(A_{\Delta}, P_{\mathrm{g}}) \leftarrow (A_{\Delta} \cup \{\mathrm{head}(r_{\mathrm{g}}) \mid r_{\mathrm{g}} \in P'_{\mathrm{g}}\}, P_{\mathrm{g}} \cup P'_{\mathrm{g}})$
            $(A_{\mathrm{n}}, A_{\mathrm{o}}, A_{\mathrm{a}}) \leftarrow (A_{\Delta} \setminus A_{\mathrm{a}}, A_{\mathrm{a}}, A_{\Delta} \cup A_{\mathrm{a}})$
        until $A_{\mathrm{n}} = \emptyset$ **or** $\{r \in C \mid \mathrm{body}^{+}(r) \cap A_{\mathrm{r}} \neq \emptyset\} = \emptyset$
    return P_{g}

Example: Grounding the Second Component

$r_n(X,Z)$	$e_a(Z,Y)$	$r(X,Y)$	$A_r = \{r(X,Y)\}$	
$r(v_1,v_2) \rightarrow$	$e(v_2,v_3) \rightarrow$	$r(v_1,v_3)$	$A_o^1 = \emptyset$	**1**
$r(v_1,v_3) \rightarrow$	$e(v_3,v_4) \rightarrow$	$r(v_1,v_4)$	$A_n^1 = \left\{ \begin{array}{l} e(v_1,v_2), r(v_1,v_2), \\ e(v_1,v_3), r(v_1,v_3), \\ e(v_2,v_3), r(v_2,v_3), \\ e(v_3,v_4), r(v_3,v_4), \ldots \end{array} \right\}$	
$r(v_2,v_3) \rightarrow$	$e(v_3,v_4) \rightarrow$	$r(v_2,v_4)$		
$r(v_3,v_4) \longrightarrow \times$			$A_a^1 = A_n^1$	
			$A_f^1 = A_n^1$	
$r(v_1,v_3) \rightarrow$	$e(v_3,v_4) \rightarrow$	$r(v_1,v_4)$	$A_o^2 = A_a^1$	**2**
$r(v_1,v_4) \longrightarrow \times$			$A_n^2 = \left\{ \begin{array}{l} r(v_1,v_3), \\ r(v_1,v_4), \\ r(v_2,v_4) \end{array} \right\}$	
$r(v_2,v_4) \longrightarrow \times$			$A_a^2 = A_a^1 \cup A_n^2$	
			$A_f^2 = A_f^1$	

where $r = reach$ and $e = edge$

Example: Grounding the Second Component

$r_{\mathrm{n}}(X, Z)$	$e_{\mathrm{a}}(Z, Y)$	$r(X, Y)$	$A_{\mathrm{r}} = \{r(X, Y)\}$	
$r(v_1, v_2)$ →	$e(v_2, v_3)$ →	$r(v_1, v_3)$	$A_{\mathrm{o}}^1 = \emptyset$	**1**
$r(v_1, v_3)$ →	$e(v_3, v_4)$ →	$r(v_1, v_4)$	$A_{\mathrm{n}}^1 = \{e(v_1, v_2), r(v_1, v_2), e(v_1, v_3), r(v_1, v_3), e(v_2, v_3), r(v_2, v_3), e(v_3, v_4), r(v_3, v_4), \ldots\}$	
$r(v_2, v_3)$ →	$e(v_3, v_4)$ →	$r(v_2, v_4)$	$A_{\mathrm{a}}^1 = A_{\mathrm{n}}^1$	
$r(v_3, v_4)$ ⟶ ×			$A_{\mathrm{f}}^1 = A_{\mathrm{n}}^1$	
$r(v_1, v_3)$ →	$e(v_3, v_4)$ →	$r(v_1, v_4)$	$A_{\mathrm{o}}^2 = A_{\mathrm{a}}^1$	**2**
$r(v_1, v_4)$ ⟶ ×			$A_{\mathrm{n}}^2 = \{r(v_1, v_3), r(v_1, v_4), r(v_2, v_4)\}$	
$r(v_2, v_4)$ ⟶ ×			$A_{\mathrm{a}}^2 = A_{\mathrm{a}}^1 \cup A_{\mathrm{n}}^2$	
			$A_{\mathrm{f}}^2 = A_{\mathrm{f}}^1$	

where r = *reach* and e = *edge*

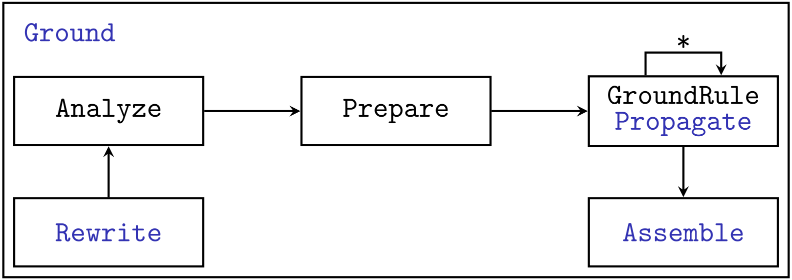
Example: Grounding the Second Component

$r_{\mathrm{n}}(X,Z)$	$e_{\mathrm{a}}(Z,Y)$	$r(X,Y)$	$A_{\mathrm{r}} = \{r(X,Y)\}$	
$r(v_1,v_2) \rightarrow$	$e(v_2,v_3) \rightarrow$	$r(v_1,v_3)$	$A_{\mathrm{o}}^1 = \emptyset$	**1**
$r(v_1,v_3) \rightarrow$	$e(v_3,v_4) \rightarrow$	$r(v_1,v_4)$	$A_{\mathrm{n}}^1 = \left\{ \begin{array}{l} e(v_1,v_2), r(v_1,v_2), \\ e(v_1,v_3), r(v_1,v_3), \\ e(v_2,v_3), r(v_2,v_3), \\ e(v_3,v_4), r(v_3,v_4), \ldots \end{array} \right\}$	
$r(v_2,v_3) \rightarrow$	$e(v_3,v_4) \rightarrow$	$r(v_2,v_4)$		
$r(v_3,v_4) \longrightarrow \times$			$A_{\mathrm{a}}^1 = A_{\mathrm{n}}^1$ $A_{\mathrm{f}}^1 = A_{\mathrm{n}}^1$	
$r(v_1,v_3) \rightarrow$	$e(v_3,v_4) \rightarrow$	$r(v_1,v_4)$	$A_{\mathrm{o}}^2 = A_{\mathrm{a}}^1$	**2**
$r(v_1,v_4) \longrightarrow \times$			$A_{\mathrm{n}}^2 = \left\{ \begin{array}{l} r(v_1,v_3), \\ r(v_1,v_4), \\ r(v_2,v_4) \end{array} \right\}$	
$r(v_2,v_4) \longrightarrow \times$			$A_{\mathrm{a}}^2 = A_{\mathrm{a}}^1 \cup A_{\mathrm{n}}^2$ $A_{\mathrm{f}}^2 = A_{\mathrm{f}}^1$	

where $r = \mathit{reach}$ and $e = \mathit{edge}$

Outline

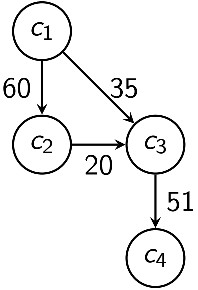
Grounding Recursive Aggregates



- grounding algorithm `Ground` is extended with three functions

`Rewrite`	rewrites aggregates into normal rules
`Propagate`	propagates aggregate atoms
`Assemble`	reconstructs aggregates

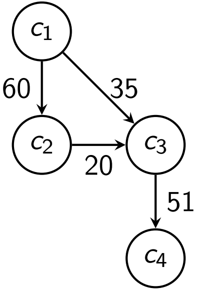
Company Controls Problem



$$
\begin{array}{ll}
company(c_1). & owns(c_1, c_2, \overline{60}). \\
company(c_2). & owns(c_1, c_3, \overline{20}). \\
company(c_3). & owns(c_2, c_3, \overline{35}). \\
company(c_4). & owns(c_3, c_4, \overline{51}).
\end{array}
$$

$$
\begin{aligned}
&controls(X, Y) \\
&\quad \leftarrow sum^{+}\{S : owns(X, Y, S); \\
&\qquad\qquad S, Z : controls(X, Z), owns(Z, Y, S)\} > \overline{50} \\
&\quad \wedge company(X) \wedge company(Y) \wedge X \neq Y
\end{aligned} \tag{3}
$$

Company Controls Problem



$company(c_1)$. $owns(c_1, c_2, \overline{60})$.

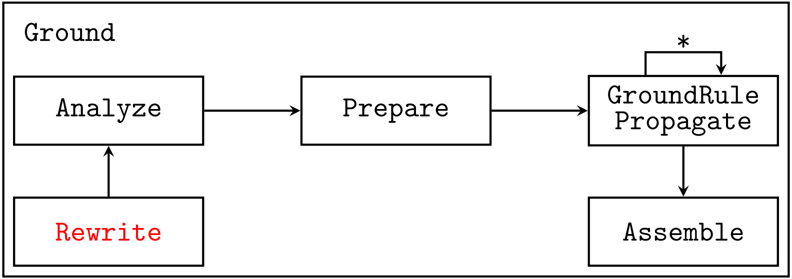
$company(c_2)$. $owns(c_1, c_3, \overline{20})$.

$company(c_3)$. $owns(c_2, c_3, \overline{35})$.

$company(c_4)$. $owns(c_3, c_4, \overline{51})$.

$$\begin{aligned} & controls(X, Y) \\ & \quad \leftarrow sum^{+}\{S : owns(X, Y, S); \\ & \qquad\qquad S, Z : controls(X, Z), owns(Z, Y, S)\} > \overline{50} \\ & \quad \wedge company(X) \wedge company(Y) \wedge X \neq Y \end{aligned} \tag{3}$$

Grounding Recursive Aggregates



Rewriting Logic Programs

function Rewrite(P)

  $Q \leftarrow \emptyset$

  // $\diamond \in \{\epsilon, \sim\}$ is the sign of the aggregate literal

  foreach r **in** P **with** $a \in \mathrm{body}(r)$, $a = \diamond\alpha E \prec s$ **do**

    let i be a unique identifier

      $\boldsymbol{x}$ be the global variables in a

      $B(\boldsymbol{L}) = \bigwedge_{l \in \mathrm{body}(r) \setminus \boldsymbol{L}, l \text{ is a simple literal}} l^{\dagger}$

    replace occurrence a in P with $\diamond aggr_i(\boldsymbol{x})$

    // tuple $\boldsymbol{L}$ converts to a set below

    $Q \leftarrow Q \cup \{accu_i(\boldsymbol{x}, neutral) \leftarrow \widehat{\alpha}(\emptyset) \prec s \wedge B(\emptyset)\}$

      $\cup \{accu_i(\boldsymbol{x}, tuple(\boldsymbol{t})) \leftarrow \bigwedge_{l \in \boldsymbol{L}} l \wedge B(\boldsymbol{L}) \mid \boldsymbol{t} : \boldsymbol{L} \in E\}$

      $\cup \{aggr_i(\boldsymbol{x}) \leftarrow accu_i(\boldsymbol{x}, \_) \wedge \bot\}$

  return $P \cup Q$

Rewriting Logic Programs

function `Rewrite`(P)

  $Q \leftarrow \emptyset$

  `// `$\diamond \in \{\epsilon, \sim\}$` is the sign of the aggregate literal`

  foreach r **in** P **with** $a \in \mathrm{body}(r)$, $a = \diamond\alpha E \prec s$ **do**

    let i be a unique identifier

      $\boldsymbol{x}$ be the global variables in a

      $B(\boldsymbol{L}) = \bigwedge_{l \in \mathrm{body}(r) \setminus \boldsymbol{L}, l \text{ is a simple literal}} l^{\dagger}$

    replace occurrence a in P with $\diamond aggr_i(\boldsymbol{x})$

    `// tuple `$\boldsymbol{L}$` converts to a set below`

    $Q \leftarrow Q \cup \{accu_i(\boldsymbol{x}, neutral) \leftarrow \widehat{\alpha}(\emptyset) \prec s \wedge B(\emptyset)\}$

      $\cup \{accu_i(\boldsymbol{x}, tuple(\boldsymbol{t})) \leftarrow \bigwedge_{l \in \boldsymbol{L}} l \wedge B(\boldsymbol{L}) \mid \boldsymbol{t} : \boldsymbol{L} \in E\}$

      $\cup \{aggr_i(\boldsymbol{x}) \leftarrow accu_i(\boldsymbol{x}, \_) \wedge \bot\}$

  return $P \cup Q$

Rewriting Logic Programs

function `Rewrite`(P)

$Q \leftarrow \emptyset$

`// ◇ ∈ {ϵ, ∼} is the sign of the aggregate literal`

foreach r **in** P **with** $a \in \mathrm{body}(r)$, $a = \diamond \alpha E \prec s$ **do**

let i be a unique identifier

$\boldsymbol{x}$ be the global variables in a

$B(\boldsymbol{L}) = \bigwedge_{l \in \mathrm{body}(r) \setminus \boldsymbol{L}, l \text{ is a simple literal}} l^{\dagger}$

replace occurrence a in P with $\diamond aggr_i(\boldsymbol{x})$

`// tuple L converts to a set below`

$Q \leftarrow Q \cup \{accu_i(\boldsymbol{x}, neutral) \leftarrow \widehat{\alpha}(\emptyset) \prec s \wedge B(\emptyset)\}$

$\cup \{accu_i(\boldsymbol{x}, tuple(\boldsymbol{t})) \leftarrow \bigwedge_{l \in \boldsymbol{L}} l \wedge B(\boldsymbol{L}) \mid \boldsymbol{t} : \boldsymbol{L} \in E\}$

$\cup \{aggr_i(\boldsymbol{x}) \leftarrow accu_i(\boldsymbol{x}, \_) \wedge \bot\}$

return $P \cup Q$

Company Controls Rewritten

$$controls(X, Y) \leftarrow aggr_1(X, Y) \wedge B \tag{4}$$

$$accu_1(X, Y, neutral) \leftarrow \overline{0} > \overline{50} \wedge B^\dagger \tag{5}$$

$$accu_1(X, Y, tuple(S)) \leftarrow owns(X, Y, S) \wedge B^\dagger \tag{6}$$

$$accu_1(X, Y, tuple(S, Z)) \leftarrow controls(X, Z) \wedge owns(Z, Y, S) \wedge B^\dagger \tag{7}$$

$$aggr_1(X, Y) \leftarrow accu_1(X, Y, \_) \wedge \bot \tag{8}$$

where $B = company(X) \wedge company(Y) \wedge X \neq Y$

Company Controls Analyze and Prepared

Component$_1$: $accu_1(X, Y, neutral) \leftarrow \overline{0} > \overline{50} \wedge B^{\dagger}_{\mathrm{n}}$

Component$_2$: $accu_1(X, Y, tuple(S)) \leftarrow owns_{\mathrm{n}}(X, Y, S) \wedge B^{\dagger}_{\mathrm{n}}$

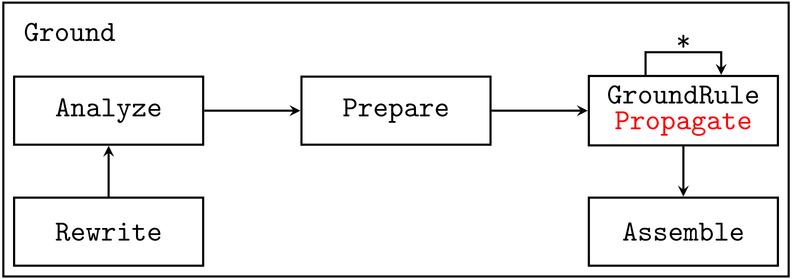
Component$_3$: $aggr_1(X, Y) \leftarrow \mathbf{accu_1}_{\mathrm{n}}(\mathbf{X}, \mathbf{Y}, \_) \wedge \bot$

$controls(X, Y) \leftarrow \mathbf{aggr_1}_{\mathrm{n}}(\mathbf{X}, \mathbf{Y}) \wedge B_{\mathrm{a}}$

$accu_1(X, Y, tuple(S, Z)) \leftarrow \mathbf{controls}_{\mathrm{n}}(\mathbf{X}, \mathbf{Z}) \wedge owns_{\mathrm{a}}(Z, Y, S) \wedge B^{\dagger}_{\mathrm{a}}$

where $B_x = company_x(X) \wedge company_x(Y) \wedge X \neq Y$

Grounding Recursive Aggregates



Propagating Aggregates

function $\texttt{Propagate}(I, r, A_{\mathrm{a}}, A_{\mathrm{f}})$

  $A_{\Delta} \leftarrow \emptyset$

  foreach $i, \boldsymbol{g}$ **where** $i \in I$ **and** $accu_i(\boldsymbol{g}, t) \in A_{\mathrm{a}}$ **do**

    let $T_{\mathrm{f}} = \{\boldsymbol{t} \mid accu_i(\boldsymbol{g}, tuple(\boldsymbol{t})) \in A_{\mathrm{f}}, \boldsymbol{t}$ is relevant for $\alpha_i\}$

      $T_{\mathrm{a}} = \{\boldsymbol{t} \mid accu_i(\boldsymbol{g}, tuple(\boldsymbol{t})) \in A_{\mathrm{a}}, \boldsymbol{t}$ is relevant for $\alpha_i\}$

    if exists $T_{\mathrm{f}} \subseteq T \subseteq T_{\mathrm{a}}$ **where** $\widehat{\alpha}_i(T) \prec_i (s_i)_{\boldsymbol{g}}^{\boldsymbol{x}_i}$ *is true* **then**

      if (*aggregate* i *is monotone* **and** $\widehat{\alpha}_i(T_{\mathrm{f}}) \prec_i (s_i)_{\boldsymbol{g}}^{\boldsymbol{x}_i}$)

        or (**not** r **and** $T_{\mathrm{a}} \setminus T_{\mathrm{f}} = \emptyset$) **then**

        $A_{\mathrm{f}} \leftarrow A_{\mathrm{f}} \cup \{aggr_i(\boldsymbol{g})\}$

      $A_{\Delta} \leftarrow A_{\Delta} \cup \{aggr_i(\boldsymbol{g})\}$

  return $(A_{\Delta}, A_{\mathrm{f}})$

Propagating Aggregates

function Propagate$(I, r, A_{\mathrm{a}}, A_{\mathrm{f}})$

$A_\Delta \leftarrow \emptyset$

foreach $i, \boldsymbol{g}$ **where** $i \in I$ **and** $accu_i(\boldsymbol{g}, t) \in A_{\mathrm{a}}$ **do**

let $T_{\mathrm{f}} = \{\boldsymbol{t} \mid accu_i(\boldsymbol{g}, tuple(\boldsymbol{t})) \in A_{\mathrm{f}}, \boldsymbol{t}$ is relevant for $\alpha_i\}$

$T_{\mathrm{a}} = \{\boldsymbol{t} \mid accu_i(\boldsymbol{g}, tuple(\boldsymbol{t})) \in A_{\mathrm{a}}, \boldsymbol{t}$ is relevant for $\alpha_i\}$

if exists $T_{\mathrm{f}} \subseteq T \subseteq T_{\mathrm{a}}$ **where** $\widehat{\alpha}_i(T) \prec_i (s_i)_{\boldsymbol{g}}^{\boldsymbol{x}_i}$ *is true* **then**

if (*aggregate* i *is monotone* **and** $\widehat{\alpha}_i(T_{\mathrm{f}}) \prec_i (s_i)_{\boldsymbol{g}}^{\boldsymbol{x}_i}$)

or (**not** r **and** $T_{\mathrm{a}} \setminus T_{\mathrm{f}} = \emptyset$) **then**

$A_{\mathrm{f}} \leftarrow A_{\mathrm{f}} \cup \{aggr_i(\boldsymbol{g})\}$

$A_\Delta \leftarrow A_\Delta \cup \{aggr_i(\boldsymbol{g})\}$

return $(A_\Delta, A_{\mathrm{f}})$

Propagating Aggregates

function Propagate$(I, r, A_{\mathrm{a}}, A_{\mathrm{f}})$
  $A_\Delta \leftarrow \emptyset$
  foreach $i, \boldsymbol{g}$ **where** $i \in I$ **and** $accu_i(\boldsymbol{g}, t) \in A_{\mathrm{a}}$ **do**
    let $T_{\mathrm{f}} = \{\boldsymbol{t} \mid accu_i(\boldsymbol{g}, tuple(\boldsymbol{t})) \in A_{\mathrm{f}}, \boldsymbol{t} \text{ is relevant for } \alpha_i\}$
      $T_{\mathrm{a}} = \{\boldsymbol{t} \mid accu_i(\boldsymbol{g}, tuple(\boldsymbol{t})) \in A_{\mathrm{a}}, \boldsymbol{t} \text{ is relevant for } \alpha_i\}$
    if exists $T_{\mathrm{f}} \subseteq T \subseteq T_{\mathrm{a}}$ **where** $\widehat{\alpha}_i(T) \prec_i (s_i)_{\boldsymbol{g}}^{\boldsymbol{x}_i}$ *is true* **then**
      if (*aggregate i is monotone* **and** $\widehat{\alpha}_i(T_{\mathrm{f}}) \prec_i (s_i)_{\boldsymbol{g}}^{\boldsymbol{x}_i}$)
        or (**not** r **and** $T_{\mathrm{a}} \setminus T_{\mathrm{f}} = \emptyset$) **then**
        $A_{\mathrm{f}} \leftarrow A_{\mathrm{f}} \cup \{aggr_i(\boldsymbol{g})\}$
      $A_\Delta \leftarrow A_\Delta \cup \{aggr_i(\boldsymbol{g})\}$
  return $(A_\Delta, A_{\mathrm{f}})$

Propagating Aggregates

function $\texttt{Propagate}(I, r, A_{\mathrm{a}}, A_{\mathrm{f}})$

  $A_\Delta \leftarrow \emptyset$

  foreach $i, \boldsymbol{g}$ **where** $i \in I$ **and** $accu_i(\boldsymbol{g}, t) \in A_{\mathrm{a}}$ **do**

    let $T_{\mathrm{f}} = \{\boldsymbol{t} \mid accu_i(\boldsymbol{g}, tuple(\boldsymbol{t})) \in A_{\mathrm{f}}, \boldsymbol{t}$ is relevant for $\alpha_i\}$

      $T_{\mathrm{a}} = \{\boldsymbol{t} \mid accu_i(\boldsymbol{g}, tuple(\boldsymbol{t})) \in A_{\mathrm{a}}, \boldsymbol{t}$ is relevant for $\alpha_i\}$

    if exists $T_{\mathrm{f}} \subseteq T \subseteq T_{\mathrm{a}}$ **where** $\widehat{\alpha}_i(T) \prec_i (s_i)_{\boldsymbol{g}}^{\boldsymbol{x}_i}$ *is true* **then**

      if (*aggregate* i *is monotone* **and** $\widehat{\alpha}_i(T_{\mathrm{f}}) \prec_i (s_i)_{\boldsymbol{g}}^{\boldsymbol{x}_i}$)

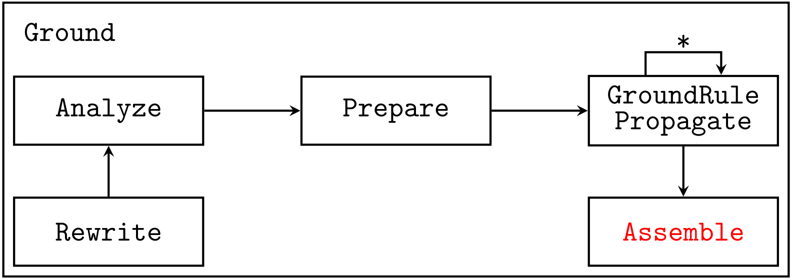
        or (**not** r **and** $T_{\mathrm{a}} \setminus T_{\mathrm{f}} = \emptyset$) **then**

        $A_{\mathrm{f}} \leftarrow A_{\mathrm{f}} \cup \{aggr_i(\boldsymbol{g})\}$

      $A_\Delta \leftarrow A_\Delta \cup \{aggr_i(\boldsymbol{g})\}$

  return $(A_\Delta, A_{\mathrm{f}})$

Grounding Recursive Aggregates



Assembling Aggregates

function `Assemble`(P_{g})
  foreach $aggr_i(\boldsymbol{g})$ *occurring in* P_{g} **do**
    `//` $\mathrm{body}(r)$ `converts to a tuple of literals`
    let $E = \{\boldsymbol{t} : \mathrm{body}(r) \mid r \in P_{\mathrm{g}}, \mathrm{head}(r) = accu_i(\boldsymbol{g}, tuple(\boldsymbol{t}))\}$
    replace occurrences of $aggr_i(\boldsymbol{g})$ in P_{g} with $\alpha_i(E) \prec_i (s_i)_{\boldsymbol{g}}^{\boldsymbol{x}_i}$
  remove all rules with atoms over $accu_i$ in the head from P_{g}
  return P_{g}

Assembling Aggregates

function `Assemble`(P_{g})

  foreach $aggr_i(\boldsymbol{g})$ *occurring in* P_{g} **do**

    `//` $\mathrm{body}(r)$ `converts to a tuple of literals`

    let $E = \{\boldsymbol{t} : \mathrm{body}(r) \mid r \in P_{\mathrm{g}}, \mathrm{head}(r) = accu_i(\boldsymbol{g}, tuple(\boldsymbol{t}))\}$

    replace occurrences of $aggr_i(\boldsymbol{g})$ in P_{g} with $\alpha_i(E) \prec_i (s_i)_{\boldsymbol{g}}^{\boldsymbol{x}_i}$

  remove all rules with atoms over $accu_i$ in the head from P_{g}

  return P_{g}

Assembling Aggregates

function `Assemble`(P_{g})

  foreach *$aggr_i(\boldsymbol{g})$ occurring in P_{g}* **do**

    // $\mathrm{body}(r)$ converts to a tuple of literals

    let $E = \{\boldsymbol{t} : \mathrm{body}(r) \mid r \in P_{\mathrm{g}}, \mathrm{head}(r) = accu_i(\boldsymbol{g}, tuple(\boldsymbol{t}))\}$

    replace occurrences of $aggr_i(\boldsymbol{g})$ in P_{g} with $\alpha_i(E) \prec_i (s_i)^{\boldsymbol{x}_i}_{\boldsymbol{g}}$

  remove all rules with atoms over $accu_i$ in the head from P_{g}

  return P_{g}

Assembling Aggregates

function Assemble(P_{g})

  foreach *$aggr_i(\boldsymbol{g})$ occurring in P_{g}* **do**

    // $\mathrm{body}(r)$ converts to a tuple of literals

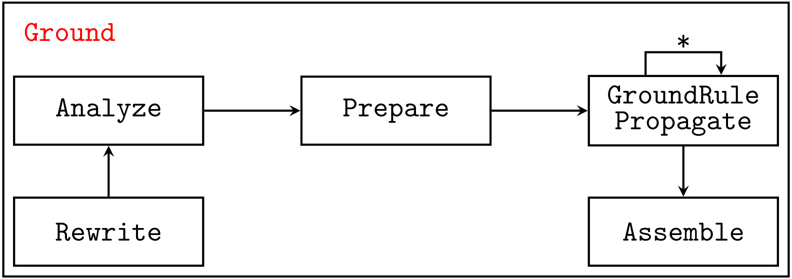
    let $E = \{\boldsymbol{t} : \mathrm{body}(r) \mid r \in P_{\mathrm{g}}, \mathrm{head}(r) = accu_i(\boldsymbol{g}, tuple(\boldsymbol{t}))\}$

    replace occurrences of $aggr_i(\boldsymbol{g})$ in P_{g} with $\alpha_i(E) \prec_i (s_i)^{\boldsymbol{x}_i}_{\boldsymbol{g}}$

  remove all rules with atoms over $accu_i$ in the head from P_{g}

  return P_{g}

Grounding Recursive Aggregates



Grounding Recursive Aggregates

function $\texttt{Ground}(P, A_\mathrm{f})$
    $(P_\mathrm{g}, A_\mathrm{a}) \leftarrow (\emptyset, A_\mathrm{f})$
    foreach (C, A_r) **in** $\texttt{Analyze}(\texttt{Rewrite}(P))$ **do**
        let $I = \{i \mid \mathit{aggr}_i$ occurs in a rule head in $C\}$
            $I_\mathrm{r} = \{i \mid r \in C, \mathrm{head}(r) = \mathit{accu}_i(\boldsymbol{x}, t), a \in \mathrm{body}^+(r) \cap A_\mathrm{r}, r \not\succ a\}$
        $(A_\mathrm{n}, A_\mathrm{o}) \leftarrow (A_\mathrm{a}, \emptyset)$
        repeat
            $A_\Delta \leftarrow \emptyset$
            foreach r **in** $\texttt{Prepare}(C, A_\mathrm{r})$ **do**
                $(P'_\mathrm{g}, A_\mathrm{f}) \leftarrow \texttt{GroundRule}(r, A_\mathrm{r}, A_\mathrm{n}, A_\mathrm{o}, A_\mathrm{a}, A_\mathrm{f})$
                $(A_\Delta, P_\mathrm{g}) \leftarrow (A_\Delta \cup \{\mathrm{head}(r_\mathrm{g}) \mid r_\mathrm{g} \in P'_\mathrm{g}\}, P_\mathrm{g} \cup P'_\mathrm{g})$
            if $A_\Delta \subseteq A_\mathrm{a}$ **then**
                $(A_\Delta, A_\mathrm{f}) \leftarrow \texttt{Propagate}(I \setminus I_\mathrm{r}, \textbf{false}, A_\mathrm{a}, A_\mathrm{f})$
            if $A_\Delta \subseteq A_\mathrm{a}$ **then**
                $(A_\Delta, A_\mathrm{f}) \leftarrow \texttt{Propagate}(I \cap I_\mathrm{r}, \textbf{true}, A_\mathrm{a}, A_\mathrm{f})$
            $(A_\mathrm{n}, A_\mathrm{o}, A_\mathrm{a}) \leftarrow (A_\Delta \setminus A_\mathrm{a}, A_\mathrm{a}, A_\Delta \cup A_\mathrm{a})$
        until $A_\mathrm{n} = \emptyset$ **or** $\{r \in C \mid \mathrm{body}^+(r) \cap A_\mathrm{r} \neq \emptyset\} = \emptyset$
    return $\texttt{Assemble}(P_\mathrm{g})$

Grounding Recursive Aggregates

function $\texttt{Ground}(P, A_\mathrm{f})$
    $(P_\mathrm{g}, A_\mathrm{a}) \leftarrow (\emptyset, A_\mathrm{f})$
    foreach (C, A_r) **in** $\texttt{Analyze}(\texttt{Rewrite}(P))$ **do**
        let $I = \{i \mid \mathit{aggr}_i$ occurs in a rule head in $C\}$
            $I_\mathrm{r} = \{i \mid r \in C, \mathrm{head}(r) = \mathit{accu}_i(\boldsymbol{x}, t), a \in \mathrm{body}^+(r) \cap A_\mathrm{r}, r \not\prec a\}$
        $(A_\mathrm{n}, A_\mathrm{o}) \leftarrow (A_\mathrm{a}, \emptyset)$
        repeat
            $A_\Delta \leftarrow \emptyset$
            foreach r **in** $\texttt{Prepare}(C, A_\mathrm{r})$ **do**
                $(P'_\mathrm{g}, A_\mathrm{f}) \leftarrow \texttt{GroundRule}(r, A_\mathrm{r}, A_\mathrm{n}, A_\mathrm{o}, A_\mathrm{a}, A_\mathrm{f})$
                $(A_\Delta, P_\mathrm{g}) \leftarrow (A_\Delta \cup \{\mathrm{head}(r_\mathrm{g}) \mid r_\mathrm{g} \in P'_\mathrm{g}\}, P_\mathrm{g} \cup P'_\mathrm{g})$
            if $A_\Delta \subseteq A_\mathrm{a}$ **then**
                $(A_\Delta, A_\mathrm{f}) \leftarrow \texttt{Propagate}(I \setminus I_\mathrm{r}, \textbf{false}, A_\mathrm{a}, A_\mathrm{f})$
            if $A_\Delta \subseteq A_\mathrm{a}$ **then**
                $(A_\Delta, A_\mathrm{f}) \leftarrow \texttt{Propagate}(I \cap I_\mathrm{r}, \textbf{true}, A_\mathrm{a}, A_\mathrm{f})$
            $(A_\mathrm{n}, A_\mathrm{o}, A_\mathrm{a}) \leftarrow (A_\Delta \setminus A_\mathrm{a}, A_\mathrm{a}, A_\Delta \cup A_\mathrm{a})$
        until $A_\mathrm{n} = \emptyset$ **or** $\{r \in C \mid \mathrm{body}^+(r) \cap A_\mathrm{r} \neq \emptyset\} = \emptyset$
    return $\texttt{Assemble}(P_\mathrm{g})$

Grounding Recursive Aggregates

function Ground(P, A_f)
$(P_\mathrm{g}, A_\mathrm{a}) \leftarrow (\emptyset, A_\mathrm{f})$
foreach (C, A_r) **in** Analyze(Rewrite(P)) **do**
let $I = \{i \mid aggr_i$ occurs in a rule head in $C\}$
$I_\mathrm{r} = \{i \mid r \in C, \mathrm{head}(r) = accu_i(\boldsymbol{x}, t), a \in \mathrm{body}^+(r) \cap A_\mathrm{r}, r \not\prec a\}$
$(A_\mathrm{n}, A_\mathrm{o}) \leftarrow (A_\mathrm{a}, \emptyset)$
repeat
$A_\Delta \leftarrow \emptyset$
foreach r **in** Prepare(C, A_r) **do**
$(P'_\mathrm{g}, A_\mathrm{f}) \leftarrow$ GroundRule$(r, A_\mathrm{r}, A_\mathrm{n}, A_\mathrm{o}, A_\mathrm{a}, A_\mathrm{f})$
$(A_\Delta, P_\mathrm{g}) \leftarrow (A_\Delta \cup \{\mathrm{head}(r_\mathrm{g}) \mid r_\mathrm{g} \in P'_\mathrm{g}\}, P_\mathrm{g} \cup P'_\mathrm{g})$
if $A_\Delta \subseteq A_\mathrm{a}$ **then**
$(A_\Delta, A_\mathrm{f}) \leftarrow$ Propagate$(I \setminus I_\mathrm{r}, \textbf{false}, A_\mathrm{a}, A_\mathrm{f})$
if $A_\Delta \subseteq A_\mathrm{a}$ **then**
$(A_\Delta, A_\mathrm{f}) \leftarrow$ Propagate$(I \cap I_\mathrm{r}, \textbf{true}, A_\mathrm{a}, A_\mathrm{f})$
$(A_\mathrm{n}, A_\mathrm{o}, A_\mathrm{a}) \leftarrow (A_\Delta \setminus A_\mathrm{a}, A_\mathrm{a}, A_\Delta \cup A_\mathrm{a})$
until $A_\mathrm{n} = \emptyset$ **or** $\{r \in C \mid \mathrm{body}^+(r) \cap A_\mathrm{r} \neq \emptyset\} = \emptyset$
return Assemble(P_g)

Grounding Recursive Aggregates

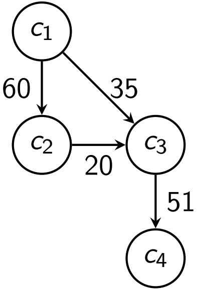
function $\texttt{Ground}(P, A_\mathrm{f})$
    $(P_\mathrm{g}, A_\mathrm{a}) \leftarrow (\emptyset, A_\mathrm{f})$
    foreach (C, A_r) **in** $\texttt{Analyze}(\texttt{Rewrite}(P))$ **do**
        let $I = \{i \mid aggr_i \text{ occurs in a rule head in } C\}$
            $I_\mathrm{r} = \{i \mid r \in C, \mathrm{head}(r) = accu_i(\boldsymbol{x}, t), a \in \mathrm{body}^+(r) \cap A_\mathrm{r}, r \not\prec a\}$
        $(A_\mathrm{n}, A_\mathrm{o}) \leftarrow (A_\mathrm{a}, \emptyset)$
        repeat
            $A_\Delta \leftarrow \emptyset$
            foreach r **in** $\texttt{Prepare}(C, A_\mathrm{r})$ **do**
                $(P'_\mathrm{g}, A_\mathrm{f}) \leftarrow \texttt{GroundRule}(r, A_\mathrm{r}, A_\mathrm{n}, A_\mathrm{o}, A_\mathrm{a}, A_\mathrm{f})$
                $(A_\Delta, P_\mathrm{g}) \leftarrow (A_\Delta \cup \{\mathrm{head}(r_\mathrm{g}) \mid r_\mathrm{g} \in P'_\mathrm{g}\}, P_\mathrm{g} \cup P'_\mathrm{g})$
            if $A_\Delta \subseteq A_\mathrm{a}$ **then**
                $(A_\Delta, A_\mathrm{f}) \leftarrow \texttt{Propagate}(I \setminus I_\mathrm{r}, \textbf{false}, A_\mathrm{a}, A_\mathrm{f})$
            if $A_\Delta \subseteq A_\mathrm{a}$ **then**
                $(A_\Delta, A_\mathrm{f}) \leftarrow \texttt{Propagate}(I \cap I_\mathrm{r}, \textbf{true}, A_\mathrm{a}, A_\mathrm{f})$
            $(A_\mathrm{n}, A_\mathrm{o}, A_\mathrm{a}) \leftarrow (A_\Delta \setminus A_\mathrm{a}, A_\mathrm{a}, A_\Delta \cup A_\mathrm{a})$
        until $A_\mathrm{n} = \emptyset$ **or** $\{r \in C \mid \mathrm{body}^+(r) \cap A_\mathrm{r} \neq \emptyset\} = \emptyset$
    return $\texttt{Assemble}(P_\mathrm{g})$

Grounding Recursive Aggregates

function $\texttt{Ground}(P, A_\mathrm{f})$
    $(P_\mathrm{g}, A_\mathrm{a}) \leftarrow (\emptyset, A_\mathrm{f})$
    foreach (C, A_r) **in** $\texttt{Analyze}(\texttt{Rewrite}(P))$ **do**
        let $I = \{i \mid \mathit{aggr}_i$ occurs in a rule head in $C\}$
            $I_\mathrm{r} = \{i \mid r \in C, \mathrm{head}(r) = \mathit{accu}_i(\boldsymbol{x}, t), a \in \mathrm{body}^+(r) \cap A_\mathrm{r}, r \not\prec a\}$
        $(A_\mathrm{n}, A_\mathrm{o}) \leftarrow (A_\mathrm{a}, \emptyset)$
        repeat
            $A_\Delta \leftarrow \emptyset$
            foreach r **in** $\texttt{Prepare}(C, A_\mathrm{r})$ **do**
                $(P'_\mathrm{g}, A_\mathrm{f}) \leftarrow \texttt{GroundRule}(r, A_\mathrm{r}, A_\mathrm{n}, A_\mathrm{o}, A_\mathrm{a}, A_\mathrm{f})$
                $(A_\Delta, P_\mathrm{g}) \leftarrow (A_\Delta \cup \{\mathrm{head}(r_\mathrm{g}) \mid r_\mathrm{g} \in P'_\mathrm{g}\}, P_\mathrm{g} \cup P'_\mathrm{g})$
            if $A_\Delta \subseteq A_\mathrm{a}$ **then**
                $(A_\Delta, A_\mathrm{f}) \leftarrow \texttt{Propagate}(I \setminus I_\mathrm{r}, \textbf{false}, A_\mathrm{a}, A_\mathrm{f})$
            if $A_\Delta \subseteq A_\mathrm{a}$ **then**
                $(A_\Delta, A_\mathrm{f}) \leftarrow \texttt{Propagate}(I \cap I_\mathrm{r}, \textbf{true}, A_\mathrm{a}, A_\mathrm{f})$
            $(A_\mathrm{n}, A_\mathrm{o}, A_\mathrm{a}) \leftarrow (A_\Delta \setminus A_\mathrm{a}, A_\mathrm{a}, A_\Delta \cup A_\mathrm{a})$
        until $A_\mathrm{n} = \emptyset$ **or** $\{r \in C \mid \mathrm{body}^+(r) \cap A_\mathrm{r} \neq \emptyset\} = \emptyset$
    return $\texttt{Assemble}(P_\mathrm{g})$

Outline

Company Controls Problem



$$
\begin{array}{ll}
company(c_1). & owns(c_1, c_2, \overline{60}). \\
company(c_2). & owns(c_1, c_3, \overline{20}). \\
company(c_3). & owns(c_2, c_3, \overline{35}). \\
company(c_4). & owns(c_3, c_4, \overline{51}).
\end{array}
$$

$$
\begin{aligned}
&controls(X, Y) \\
&\quad \leftarrow sum^{+}\{S : owns(X, Y, S); \\
&\qquad\qquad S, Z : controls(X, Z), owns(Z, Y, S)\} > \overline{50} \\
&\quad \wedge company(X) \wedge company(Y) \wedge X \neq Y
\end{aligned}
\tag{3}
$$

Example: Grounding Component$_1$

Component$_1$: $accu_1(X, Y, neutral) \leftarrow \overline{0} > \overline{50} \wedge B_{\mathrm{n}}^{\dagger}$

Component$_2$: $accu_1(X, Y, tuple(S)) \leftarrow owns_{\mathrm{n}}(X, Y, S) \wedge B_{\mathrm{n}}^{\dagger}$

Component$_3$: $aggr_1(X, Y) \leftarrow \mathbf{accu_1}_{\mathrm{n}}(\mathbf{X}, \mathbf{Y}, \_) \wedge \bot$

$controls(X, Y) \leftarrow \mathbf{aggr_1}_{\mathrm{n}}(\mathbf{X}, \mathbf{Y}) \wedge B_{\mathrm{a}}$

$accu_1(X, Y, tuple(S, Z)) \leftarrow \mathbf{controls}_{\mathrm{n}}(\mathbf{X}, \mathbf{Z}) \wedge owns_{\mathrm{a}}(Z, Y, S) \wedge B_{\mathrm{a}}^{\dagger}$

where $B_x = company_x(X) \wedge company_x(Y) \wedge X \neq Y$

Example: Grounding Component$_1$

$$\frac{\overline{0} > \overline{50} \qquad c_{\mathrm{n}}^{\dagger}(X) \qquad c_{\mathrm{n}}^{\dagger}(Y) \qquad X \neq Y \qquad\qquad a_1(X, Y, n)}{\times \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathbf{1}}$$

where $a_1 = accu_1$, $g_1 = aggr_1$, $r = controls$, $c = company$, $o = owns$, $n = neutral$, $t = tuple$

Example: Grounding Component$_2$

Component$_1$: $accu_1(X, Y, neutral) \leftarrow \overline{0} > \overline{50} \wedge B_{\mathrm{n}}^{\dagger}$

Component$_2$: $accu_1(X, Y, tuple(S)) \leftarrow owns_{\mathrm{n}}(X, Y, S) \wedge B_{\mathrm{n}}^{\dagger}$

Component$_3$: $aggr_1(X, Y) \leftarrow \mathbf{accu_1}_{\mathrm{n}}(\mathbf{X}, \mathbf{Y}, \_) \wedge \bot$

$controls(X, Y) \leftarrow \mathbf{aggr_1}_{\mathrm{n}}(\mathbf{X}, \mathbf{Y}) \wedge B_{\mathrm{a}}$

$accu_1(X, Y, tuple(S, Z)) \leftarrow \mathbf{controls}_{\mathrm{n}}(\mathbf{X}, \mathbf{Z}) \wedge owns_{\mathrm{a}}(Z, Y, S) \wedge B_{\mathrm{a}}^{\dagger}$

where $B_x = company_x(X) \wedge company_x(Y) \wedge X \neq Y$

Example: Grounding Component$_2$

$o_{\mathrm{n}}(X, Y, S)$		$c^{\dagger}_{\mathrm{n}}(X)$		$c^{\dagger}_{\mathrm{n}}(Y)$		$X \neq Y$		$a_1(X, Y, t(S))$	
$o(c_1, c_2, \overline{60})$	$\longrightarrow$	$c(c_1)$	$\longrightarrow$	$c(c_2)$	$\rightarrow$	$c_1 \neq c_2$	$\longrightarrow$	$a_1(c_1, c_2, t(\overline{60}))$	1
$o(c_1, c_3, \overline{20})$	$\longrightarrow$	$c(c_1)$	$\longrightarrow$	$c(c_3)$	$\rightarrow$	$c_1 \neq c_3$	$\longrightarrow$	$a_1(c_1, c_3, t(\overline{20}))$	
$o(c_2, c_3, \overline{35})$	$\longrightarrow$	$c(c_2)$	$\longrightarrow$	$c(c_3)$	$\rightarrow$	$c_2 \neq c_3$	$\longrightarrow$	$a_1(c_2, c_3, t(\overline{35}))$	
$o(c_3, c_4, \overline{51})$	$\longrightarrow$	$c(c_3)$	$\longrightarrow$	$c(c_4)$	$\rightarrow$	$c_3 \neq c_4$	$\longrightarrow$	$a_1(c_3, c_4, t(\overline{51}))$	

where $a_1 = accu_1$, $g_1 = aggr_1$, $r = controls$, $c = company$, $o = owns$, $n = neutral$, $t = tuple$

Example: Grounding Component$_2$

$o_{\mathrm{n}}(X, Y, S)$	$c^{\dagger}_{\mathrm{n}}(X)$	$c^{\dagger}_{\mathrm{n}}(Y)$	$X \neq Y$	$a_1(X, Y, t(S))$	
$o(c_1, c_2, \overline{60})$	$c(c_1)$	$c(c_2)$	$c_1 \neq c_2$	$a_1(c_1, c_2, t(\overline{60}))$	**1**
$o(c_1, c_3, \overline{20})$	$c(c_1)$	$c(c_3)$	$c_1 \neq c_3$	$a_1(c_1, c_3, t(\overline{20}))$	
$o(c_2, c_3, \overline{35})$	$c(c_2)$	$c(c_3)$	$c_2 \neq c_3$	$a_1(c_2, c_3, t(\overline{35}))$	
$o(c_3, c_4, \overline{51})$	$c(c_3)$	$c(c_4)$	$c_3 \neq c_4$	$a_1(c_3, c_4, t(\overline{51}))$	

where $a_1 = accu_1$, $g_1 = aggr_1$, $r = controls$, $c = company$, $o = owns$, $n = neutral$, $t = tuple$

Example: Grounding Component$_2$

$o_{\mathrm{n}}(X, Y, S)$		$c_{\mathrm{n}}^{\dagger}(X)$		$c_{\mathrm{n}}^{\dagger}(Y)$		$X \neq Y$		$a_1(X, Y, t(S))$	
$o(c_1, c_2, \overline{60})$	$\longrightarrow$	$c(c_1)$	$\longrightarrow$	$c(c_2)$	$\rightarrow$	$c_1 \neq c_2$	$\longrightarrow$	$a_1(c_1, c_2, t(\overline{60}))$	**1**
$o(c_1, c_3, \overline{20})$	$\longrightarrow$	$c(c_1)$	$\longrightarrow$	$c(c_3)$	$\rightarrow$	$c_1 \neq c_3$	$\longrightarrow$	$a_1(c_1, c_3, t(\overline{20}))$	
$o(c_2, c_3, \overline{35})$	$\longrightarrow$	$c(c_2)$	$\longrightarrow$	$c(c_3)$	$\rightarrow$	$c_2 \neq c_3$	$\longrightarrow$	$a_1(c_2, c_3, t(\overline{35}))$	
$o(c_3, c_4, \overline{51})$	$\longrightarrow$	$c(c_3)$	$\longrightarrow$	$c(c_4)$	$\rightarrow$	$c_3 \neq c_4$	$\longrightarrow$	$a_1(c_3, c_4, t(\overline{51}))$	

where $a_1 = accu_1$, $g_1 = aggr_1$, $r = controls$, $c = company$, $o = owns$, $n = neutral$, $t = tuple$

Example: Grounding Component$_3$

Component$_1$: $\mathit{accu}_1(X, Y, \mathit{neutral}) \leftarrow \overline{0} > \overline{50} \wedge B_{\mathrm{n}}^{\dagger}$

Component$_2$: $\mathit{accu}_1(X, Y, \mathit{tuple}(S)) \leftarrow \mathit{owns}_{\mathrm{n}}(X, Y, S) \wedge B_{\mathrm{n}}^{\dagger}$

Component$_3$:

$\mathit{aggr}_1(X, Y) \leftarrow \mathbf{accu_1}_{\mathrm{n}}(\mathbf{X}, \mathbf{Y}, \_) \wedge \bot$

$\mathit{controls}(X, Y) \leftarrow \mathbf{aggr_1}_{\mathrm{n}}(\mathbf{X}, \mathbf{Y}) \wedge B_{\mathrm{a}}$

$\mathit{accu}_1(X, Y, \mathit{tuple}(S, Z)) \leftarrow \mathbf{controls}_{\mathrm{n}}(\mathbf{X}, \mathbf{Z}) \wedge \mathit{owns}_{\mathrm{a}}(Z, Y, S) \wedge B_{\mathrm{a}}^{\dagger}$

where $B_x = \mathit{company}_x(X) \wedge \mathit{company}_x(Y) \wedge X \neq Y$

Example: Grounding Component$_3$ Steps 1–3

$\bot$	${a_1}_{\mathrm{n}}(X, Y, \_)$				$g_1(X, Y)$
${g_1}_{\mathrm{n}}(X, Y)$	$c_{\mathrm{a}}^{\dagger}(X)$	$c_{\mathrm{a}}^{\dagger}(Y)$	$X \neq Y$		$r(X, Y)$
$r_{\mathrm{n}}(X, Z)$	$o_{\mathrm{a}}(Z, Y, S)$	$c_{\mathrm{a}}^{\dagger}(X)$	$c_{\mathrm{a}}^{\dagger}(Y)$	$X \neq Y$	$a_1(X, Y, t(S, Z))$

Propagate: $\{g_1(c_1, c_2), g_1(c_3, c_4)\}$ 1

$\times$ 2

$g_1(c_1, c_2) \longrightarrow c(c_1) \longrightarrow c(c_2) \rightarrow c_1 \neq c_2 \longrightarrow r(c_1, c_2)$

$g_1(c_3, c_4) \longrightarrow c(c_3) \longrightarrow c(c_4) \rightarrow c_3 \neq c_4 \longrightarrow r(c_3, c_4)$

$\times$

$\times$ 3

$\times$

$r(c_1, c_2) \rightarrow o(c_2, c_3, \overline{35}) \rightarrow c(c_1) \longrightarrow c(c_3) \rightarrow c_1 \neq c_3 \rightarrow a_1(c_1, c_3, t(\overline{35}, c_2))$

$r(c_3, c_4) \longrightarrow \times$

where $a_1 = accu_1$, $g_1 = aggr_1$, $r = controls$, $c = company$, $o = owns$, $n = neutral$, $t = tuple$

Example: Grounding Component$_3$ Steps 1–3

$\bot \quad {a_1}_{\mathrm{n}}(X, Y, \_) \quad g_1(X, Y)$

$g_{1\mathrm{n}}(X, Y) \quad c_{\mathrm{a}}^{\dagger}(X) \quad c_{\mathrm{a}}^{\dagger}(Y) \quad X \neq Y \quad r(X, Y)$

$r_{\mathrm{n}}(X, Z) \quad o_{\mathrm{a}}(Z, Y, S) \quad c_{\mathrm{a}}^{\dagger}(X) \quad c_{\mathrm{a}}^{\dagger}(Y) \quad X \neq Y \quad a_1(X, Y, t(S, Z))$

`Propagate:` $\{g_1(c_1, c_2), g_1(c_3, c_4)\}$ **1**

$\times$ **2**

$g_1(c_1, c_2) \longrightarrow c(c_1) \longrightarrow c(c_2) \rightarrow c_1 \neq c_2 \longrightarrow r(c_1, c_2)$

$g_1(c_3, c_4) \longrightarrow c(c_3) \longrightarrow c(c_4) \rightarrow c_3 \neq c_4 \longrightarrow r(c_3, c_4)$

$\times$

$\times$ **3**

$\times$

$r(c_1, c_2) \rightarrow o(c_2, c_3, \overline{35}) \rightarrow c(c_1) \longrightarrow c(c_3) \rightarrow c_1 \neq c_3 \rightarrow a_1(c_1, c_3, t(\overline{35}, c_2))$

$r(c_3, c_4) \longrightarrow \times$

where $a_1 = accu_1$, $g_1 = aggr_1$, $r = controls$, $c = company$, $o = owns$, $n = neutral$, $t = tuple$

Example: Grounding Component$_3$ Steps 1–3

$\bot \quad a_{1\mathrm{n}}(X, Y, \_) \quad g_1(X, Y)$

$g_{1\mathrm{n}}(X, Y) \quad c_{\mathrm{a}}^{\dagger}(X) \quad c_{\mathrm{a}}^{\dagger}(Y) \quad X \neq Y \quad r(X, Y)$

$r_{\mathrm{n}}(X, Z) \quad o_{\mathrm{a}}(Z, Y, S) \quad c_{\mathrm{a}}^{\dagger}(X) \quad c_{\mathrm{a}}^{\dagger}(Y) \quad X \neq Y \quad a_1(X, Y, t(S, Z))$

Propagate: $\{g_1(c_1, c_2), g_1(c_3, c_4)\}$ **1**

$\times$ **2**

$g_1(c_1, c_2) \longrightarrow c(c_1) \longrightarrow c(c_2) \rightarrow c_1 \neq c_2 \longrightarrow r(c_1, c_2)$

$g_1(c_3, c_4) \longrightarrow c(c_3) \longrightarrow c(c_4) \rightarrow c_3 \neq c_4 \longrightarrow r(c_3, c_4)$

$\times$

$\times$ **3**

$\times$

$r(c_1, c_2) \rightarrow o(c_2, c_3, \overline{35}) \rightarrow c(c_1) \longrightarrow c(c_3) \rightarrow c_1 \neq c_3 \rightarrow a_1(c_1, c_3, t(\overline{35}, c_2))$

$r(c_3, c_4) \longrightarrow \times$

where $a_1 = accu_1$, $g_1 = aggr_1$, $r = controls$, $c = company$, $o = owns$, $n = neutral$, $t = tuple$

Example: Grounding Component$_3$ Steps 1–3

$\bot \quad {a_1}_{\mathrm{n}}(X, Y, \_) \qquad g_1(X, Y)$

$g_{1\,\mathrm{n}}(X, Y) \quad c_{\mathrm{a}}^{\dagger}(X) \quad c_{\mathrm{a}}^{\dagger}(Y) \quad X \neq Y \qquad r(X, Y)$

$r_{\mathrm{n}}(X, Z) \quad o_{\mathrm{a}}(Z, Y, S) \quad c_{\mathrm{a}}^{\dagger}(X) \quad c_{\mathrm{a}}^{\dagger}(Y) \quad X \neq Y \quad a_1(X, Y, t(S, Z))$

Propagate: $\{g_1(c_1, c_2), g_1(c_3, c_4)\}$ **1**

$\times$ **2**

$g_1(c_1, c_2) \longrightarrow c(c_1) \longrightarrow c(c_2) \rightarrow c_1 \neq c_2 \longrightarrow r(c_1, c_2)$

$g_1(c_3, c_4) \longrightarrow c(c_3) \longrightarrow c(c_4) \rightarrow c_3 \neq c_4 \longrightarrow r(c_3, c_4)$

$\times$

$\times$ **3**

$\times$

$r(c_1, c_2) \rightarrow o(c_2, c_3, \overline{35}) \rightarrow c(c_1) \longrightarrow c(c_3) \rightarrow c_1 \neq c_3 \rightarrow a_1(c_1, c_3, t(\overline{35}, c_2))$

$r(c_3, c_4) \longrightarrow \times$

where $a_1 = accu_1$, $g_1 = aggr_1$, $r = controls$, $c = company$, $o = owns$, $n = neutral$, $t = tuple$

Example: Grounding Component$_3$ Steps 1–3

$\bot \quad a_{1\mathrm{n}}(X, Y, \_) \quad g_1(X, Y)$

- - - - - - - - - - - - - - - - - - - -

$g_{1\mathrm{n}}(X, Y) \quad c_{\mathrm{a}}^{\dagger}(X) \quad c_{\mathrm{a}}^{\dagger}(Y) \quad X \neq Y \quad r(X, Y)$

- - - - - - - - - - - - - - - - - - - -

$r_{\mathrm{n}}(X, Z) \quad o_{\mathrm{a}}(Z, Y, S) \quad c_{\mathrm{a}}^{\dagger}(X) \quad c_{\mathrm{a}}^{\dagger}(Y) \quad X \neq Y \quad a_1(X, Y, t(S, Z))$

`Propagate`: $\{g_1(c_1, c_2), g_1(c_3, c_4)\}$ **1**

$\times$ **2**

- - - - - - - - - - - - - - - - - - - -

$g_1(c_1, c_2) \longrightarrow c(c_1) \longrightarrow c(c_2) \rightarrow c_1 \neq c_2 \longrightarrow r(c_1, c_2)$

$g_1(c_3, c_4) \longrightarrow c(c_3) \longrightarrow c(c_4) \rightarrow c_3 \neq c_4 \longrightarrow r(c_3, c_4)$

- - - - - - - - - - - - - - - - - - - -

$\times$

$\times$ **3**

- - - - - - - - - - - - - - - - - - - -

$\times$

- - - - - - - - - - - - - - - - - - - -

$r(c_1, c_2) \rightarrow o(c_2, c_3, \overline{35}) \rightarrow c(c_1) \longrightarrow c(c_3) \rightarrow c_1 \neq c_3 \rightarrow a_1(c_1, c_3, t(\overline{35}, c_2))$

$r(c_3, c_4) \longrightarrow \times$

where $a_1 = accu_1$, $g_1 = aggr_1$, $r = controls$, $c = company$, $o = owns$, $n = neutral$, $t = tuple$

Example: Grounding Component$_3$ Steps 4–6

$\bot$	$a_{1\mathrm{n}}(X, Y, \_)$				$g_1(X, Y)$	
$g_{1\mathrm{n}}(X, Y)$	$c_{\mathrm{a}}^{\dagger}(X)$	$c_{\mathrm{a}}^{\dagger}(Y)$	$X \neq Y$		$r(X, Y)$	
$r_{\mathrm{n}}(X, Z)$	$o_{\mathrm{a}}(Z, Y, S)$	$c_{\mathrm{a}}^{\dagger}(X)$	$c_{\mathrm{a}}^{\dagger}(Y)$	$X \neq Y$	$a_1(X, Y, t(S, Z))$	
`Propagate`: $\{g_1(c_1, c_3)\}$						**4**
$\times$						**5**
$g_1(c_1, c_3) \longrightarrow c(c_1) \longrightarrow c(c_3) \rightarrow c_1 \neq c_3 \longrightarrow r(c_1, c_3)$						
$\times$						
$\times$						**6**
$\times$						
$r(c_1, c_3) \rightarrow o(c_3, c_4, \overline{51}) \rightarrow c(c_1) \longrightarrow c(c_4) \rightarrow c_1 \neq c_4 \rightarrow a_1(c_1, c_4, t(\overline{51}, c_3))$						

where $a_1 = accu_1$, $g_1 = aggr_1$, $r = controls$, $c = company$, $o = owns$, $n = neutral$, $t = tuple$

Example: Grounding Component$_3$ Steps 7–9

$\bot$	$a_{1\mathrm{n}}(X, Y, \_)$				$g_1(X, Y)$	
$g_{1\mathrm{n}}(X, Y)$	$c_{\mathrm{a}}^{\dagger}(X)$	$c_{\mathrm{a}}^{\dagger}(Y)$	$X \neq Y$		$r(X, Y)$	
$r_{\mathrm{n}}(X, Z)$	$o_{\mathrm{a}}(Z, Y, S)$	$c_{\mathrm{a}}^{\dagger}(X)$	$c_{\mathrm{a}}^{\dagger}(Y)$	$X \neq Y$	$a_1(X, Y, t(S, Z))$	
`Propagate:` $\{g_1(c_1, c_4)\}$						**7**
$\times$						**8**
$g_1(c_1, c_4) \longrightarrow$	$c(c_1) \longrightarrow$	$c(c_4) \rightarrow$	$c_1 \neq c_4 \longrightarrow$		$r(c_1, c_4)$	
$\times$						
$\times$						**9**
$\times$						
$r(c_1, c_4) \longrightarrow$	$\times$					

where $a_1 = accu_1$, $g_1 = aggr_1$, $r = controls$, $c = company$, $o = owns$, $n = neutral$, $t = tuple$

Example: Resulting Grounding w/o Instance

$$a_1(c_1, c_2, t(\overline{60})) \leftarrow o(c_1, c_2, \overline{60}) \land c(c_1) \land c(c_2) \land c_1 \neq c_2 \quad (9)$$

$$a_1(c_1, c_3, t(\overline{20})) \leftarrow o(c_1, c_3, \overline{20}) \land c(c_1) \land c(c_3) \land c_1 \neq c_3 \quad (10)$$

$$a_1(c_2, c_3, t(\overline{35})) \leftarrow o(c_2, c_3, \overline{35}) \land c(c_2) \land c(c_3) \land c_2 \neq c_3 \quad (11)$$

$$a_1(c_3, c_4, t(\overline{51})) \leftarrow o(c_3, c_4, \overline{51}) \land c(c_3) \land c(c_4) \land c_3 \neq c_4 \quad (12)$$

$$r(c_1, c_2) \leftarrow g_1(c_1, c_2) \land c(c_1) \land c(c_2) \land c_1 \neq c_2 \quad (13)$$

$$r(c_3, c_4) \leftarrow g_1(c_3, c_4) \land c(c_3) \land c(c_4) \land c_3 \neq c_4 \quad (14)$$

$$a_1(c_1, c_3, t(\overline{35}, c_2)) \leftarrow r(c_1, c_2) \land o(c_2, c_3, \overline{35}) \land c(c_1) \land c(c_3) \land c_1 \neq c_3 \quad (15)$$

$$r(c_1, c_3) \leftarrow g_1(c_1, c_3) \land c(c_1) \land c(c_3) \land c_1 \neq c_3 \quad (16)$$

$$a_1(c_1, c_4, t(\overline{51}, c_3)) \leftarrow r(c_1, c_3) \land o(c_3, c_4, \overline{51}) \land c(c_1) \land c(c_4) \land c_1 \neq c_4 \quad (17)$$

$$r(c_1, c_4) \leftarrow g_1(c_1, c_4) \land c(c_1) \land c(c_4) \land c_1 \neq c_4 \quad (18)$$

where $a_1 = accu_1$, $g_1 = aggr_1$, $r = controls$, $c = company$, $o = owns$, $n = neutral$, $t = tuple$

1. grounded normal logic program
2. with simplifications applied
3. with aggreagtes assembled

Example: Resulting Grounding w/o Instance

$$a_1(c_1, c_2, t(\overline{60})) \leftarrow \quad (9)$$
$$a_1(c_1, c_3, t(\overline{20})) \leftarrow \quad (10)$$
$$a_1(c_2, c_3, t(\overline{35})) \leftarrow \quad (11)$$
$$a_1(c_3, c_4, t(\overline{51})) \leftarrow \quad (12)$$
$$r(c_1, c_2) \leftarrow \quad (13)$$
$$r(c_3, c_4) \leftarrow \quad (14)$$
$$a_1(c_1, c_3, t(\overline{35}, c_2)) \leftarrow \quad (15)$$
$$r(c_1, c_3) \leftarrow \quad (16)$$
$$a_1(c_1, c_4, t(\overline{51}, c_3)) \leftarrow \quad (17)$$
$$r(c_1, c_4) \leftarrow \quad (18)$$

where $a_1 = accu_1$, $g_1 = aggr_1$, $r = controls$, $c = company$, $o = owns$, $n = neutral$, $t = tuple$

1. grounded normal logic program
2. with simplifications applied
3. with aggreagtes assembled

Example: Resulting Grounding w/o Instance

$$controls(c_1, c_2) \leftarrow \tag{13}$$

$$controls(c_3, c_4) \leftarrow \tag{14}$$

$$controls(c_1, c_3) \leftarrow \tag{16}$$

$$controls(c_1, c_4) \leftarrow \tag{18}$$

1. grounded normal logic program
2. with simplifications applied
3. with aggreagtes assembled

Outline

Conclusion

- summary
  - recursive aggregates under Ferraris' semantics
  - ground aggregates using semi-naive database evaluation
  - reduce aggregates to normal logic program
- implemented in `gringo` series 4
  - rich input language with aggregates
  - available at `http://potassco.sourceforge.net`

Conclusion

- summary
  - recursive aggregates under Ferraris' semantics
  - ground aggregates using semi-naive database evaluation
  - reduce aggregates to normal logic program
- implemented in `gringo` series 4
  - rich input language with aggregates
  - available at `http://potassco.sourceforge.net`

Conclusion

- summary
  - recursive aggregates under Ferraris' semantics
  - ground aggregates using semi-naive database evaluation
  - reduce aggregates to normal logic program
- implemented in `gringo` series 4
  - rich input language with aggregates
  - available at `http://potassco.sourceforge.net`